UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA
Departamento de Arquitectura de Computadores y Automática

EJECUCIÓN EFICIENTE DE FLUJOS DE TRABAJOS COMPUTACIONALES EN ENTORNOS GRID.

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

José Luis Vázquez Poletti

Bajo la dirección de los doctores
Ignacio Martín
Rubén Santiago

Madrid, 2009

- **ISBN: 978-84-692-1042-0**

EJECUCIÓN EFICIENTE DE FLUJOS DE TRABAJOS COMPUTACIONALES EN ENTORNOS GRID

TESIS DOCTORAL

José Luis Vázquez Poletti

Madrid, 2008

Dpto. de Arquitectura de Computadores y Automática
Universidad Complutense de Madrid

“Quien persevera gana”

Ignacio Martín Llorente

Índice

Agradecimientos I

Acerca de este Documento VII

Su questo Documento IX

About this Document XI

Resumen XV

Riepilogo XVII

Summary XIX

1. Introducción **1**

1.1. Computación Grid 1

1.2. Infraestructuras Grid 3

1.2.1. DSAGrid 4

1.2.2. GRIDIMadrid 4

1.2.3. EGEE 4

1.2.4. Otras Infraestructuras Grid 6

1.3. Tecnologías Grid 6

1.4. Introducción al Portado de Aplicaciones 9

1.5. Programación de Aplicaciones 11

1.6. Compositores de Servicios Grid y Portales Grid 14

2. Portado de Aplicaciones de Alta Productividad **17**
2.1. Física de Fusión . . . 17
2.2. De un Rayo a un Grid Computacional . . . 19
2.3. Aplicaciones de Alta Productividad . . . 20
2.4. Trazado Masivo de Rayos en Plasmas de Fusión . . . 21
2.4.1. ¿EGEE WMS o Globus GridWay? . . . 21
2.4.2. Arquitectura del Sistema . . . 26
2.5. Primeras Experiencias . . . 27
2.6. Optimización de la Ejecución . . . 32
2.7. Conclusiones . . . 34

3. Portado de Flujos de Trabajos **37**
3.1. Bioinformática . . . 38
3.2. Un Grid Computacional para comparar Proteínas . . . 40
3.3. Flujos de Trabajos . . . 41
3.4. Arquitectura del Sistema . . . 42
3.5. Primeras Experiencias . . . 43
3.6. Alternativas a la Optimización de la Ejecución . . . 47
3.6.1. Heurísticas Clásicas . . . 48
3.6.2. Heurísticas Contemporáneas . . . 49
3.6.3. Heurísticas basadas en la Tolerancia a Fallos . . . 51
3.7. Implantación de la Optimización . . . 52
3.7.1. Heurística de Replicación . . . 52
3.7.2. Resultados Experimentales . . . 53
3.7.3. Añadiendo la Aglomeración . . . 58
3.8. Modelo de Ejecución del Flujo de Trabajos . . . 59
3.8.1. Modelo Básico . . . 59
3.8.2. Modelo con Optimizaciones . . . 62
3.9. Conclusiones . . . 65

4. Principales Aportaciones y Trabajo Futuro **67**
4.1. Principales Aportaciones . . . 67
4.2. Trabajo Futuro . . . 68

5. Principali Contribuzioni e Futuri Sviluppi del Lavoro **69**
5.1. Principali Contribuzioni . . . 69
5.2. Futuri Sviluppi . . . 70

6. Principal Contributions and Future Work **71**
6.1. Principal Contributions 71
6.2. Future Work 72

A. Artículos Presentados **73**
A.1. Coordinated Harnessing of the IRISGrid and EGEE Testbeds with GridWay 73
A.2. A Comparison between Two Grid Scheduling Philosophies: EGEE WMS and GridWay 84
A.3. Execution of a Bioinformatics Application in a Joint IRISGrid/EGEE Testbed 96
A.4. A Comparative Analysis between EGEE and GridWay Workload Management Systems 105
A.5. Fusion in the Grid 115
A.6. Advanced Strategies for Efficient Workflow Management in a Protein Clustering Application with GridWay 123
A.7. Workflow Management in a Protein Clustering Application 133
A.8. Replication Heuristics for Efficient Workflow Execution on Grids . . 141
A.9. CD-HIT Workflow Execution on Grids using Replication Heuristics . 144

Índice de tablas

1.1. Infraestructura EGEE 5
1.2. Ejemplo de programación empleando SAGA 12
1.3. Ejemplo de programación empleando GridRPC 12
1.4. Ejemplo de programación empleando DRMAA 13

2.1. Recursos Grid de SWETEST VO empleados 27
2.2. Métricas de rendimiento con ambas tecnologías 28
2.3. Número de *chunks* enviados según su tamaño 34

3.1. Tamaño de archivos y número de tareas por cada división 44
3.2. Recursos de EGEE durante los primeros experimentos 45
3.3. Recursos (EGEE y GRIDIMadrid) durante los experimentos con optimización 54
3.4. Número de trabajos replanificados y tiempos medios para cada experimento con y si optimización 57
3.5. Costes de la replicación por trabajo y división de la base de datos . . 58
3.6. Tiempos de ejecución (T_{exe}) de las tareas del *camino crítico* y tamaño de los archivos 60
3.7. Análisis estadístico de los tiempos de ejecución experimentales para 20 divisiones 63

Índice de figuras

1.1. Organización de GRIDIMadrid 5
1.2. Modelo de Capas Grid 6
1.3. Componentes de Globus Toolkit 7
1.4. Arquitectura de GridWay 8
1.5. Gestión de Portlets en GridSphere 15
1.6. Creación de servicios Grid con Taverna 16

2.1. Fusión deuterio-tritio 18
2.2. Reactor Stellarator TJII del CIEMAT 19
2.3. Arquitectura del WMS de EGEE 22
2.4. Arquitectura de GridWay 23
2.5. Funcionamiento de MaRaTra 26
2.6. Infraestructura de RedIRIS 28
2.7. T_{exe} medio por centro 29
2.8. T_{xfr} medio por centro 29
2.9. Valores de r_∞ y $n_{1/2}$ para las dos plataformas 31
2.10. Rendimiento exponencial y predicho 31
2.11. DSAGrid, GRIDIMadrid y EGEE 32
2.12. Uso de *chunks* en MaRaTra 33
2.13. Tiempos totales con diferentes *chunks* 34

3.1. Algoritmo implantado por *cd-hit* 41
3.2. Tiempos medios de CPU, transferencia y espera en cola 45
3.3. Tiempos de ejecución según las divisiones realizadas 46
3.4. Número de trabajos replanificados en cada experimento 47
3.5. *Speed-up* del flujo de trabajos según las divisiones realizadas 48

3.6. Heurística de aglomeración 50
3.7. Heurística de replicación 51
3.8. Ejemplo de árbol de bloqueos en el flujo de trabajos de *cd-hit* 53
3.9. Tiempos medios consolidados con y sin optimización 55
3.10. Tiempos de ejecución del flujo de trabajos usando la replicación . . . 56
3.11. *Speed-up* de la ejecución del flujo de trabajos con optimización y sin ella 57
3.12. Tiempos del flujo de trabajos aplicando replicación y diferentes niveles de aglomeración 59
3.13. Tiempos de ejecución del flujo de trabajos y su curva de ajuste 60
3.14. Tamaño de los archivos del flujo de trabajos y su curva de ajuste . . . 61
3.15. Resultados reales de la ejecución del flujo de trabajos frente a las previstas por el modelo 62
3.16. Distibuión normal estándar 64

Agradecimientos

Este capítulo será quizás el más leído de la presente Tesis Doctoral, así que espero no decepcionar a nadie. Reconozco que, en el momento de escribir estas líneas, todavía no me creía el haber llegado hasta aquí... síntoma de que tengo mucho que agradecer y a mucha gente. Espero sinceramente no dejarme a nadie.

Primeramente, quiero agradecer a los principales responsables de que esta Tesis vea la luz, sus directores: **Ignacio Martín** y **Rubén Santiago**. Su amor por la Ciencia y su dedicación por los que empiezan en esto han sido para mí una guía constante.

Pero también hay unos responsables de que directamente yo esté aquí, y esos son mis padres, **Luciana** y **Luis**. Ellos han sabido inculcarme unos valores y el más importante, el luchar hasta el final por lo que uno quiere. Lo mismo va por **mi familia**, **tanto española como italiana**, que me han dado lo mejor de cada cultura, permitiéndome llegar a donde he llegado.

Una persona que me ha apoyado incondicionalmente desde el primer momento que nuestros destinos se entrelazaron, añadiendo más proyectos al de la presente Tesis, es **Laura**. Espero, mi *Lady Macbeth*, que la experiencia acumulada en mi Doctorado te sea de verdadera utilidad para el tuyo. Te quiero.

Por supuesto, **mi grupo de amigos con centro de operaciones en El Escorial** (y digo centro de operaciones porque no todos son de ahí) me han brindado muchísimo apoyo. Cuando en horas bajas ha llegado el fin de semana, no faltaban los planes que permitieran animarme y retomar el trabajo del lunes con una sonrisa en la cara. Sois los mejores, no tengáis la menor duda.

De forma inusual quizás, quiero terminar agradecimientos familiares expresando mi gratitud a tres seres felinos: **Socrates** y **Tebas** por un lado, que nos dejaron hace ya un un tiempo, y la jovencita **Dacya** por el otro, que entró en mi vida cuando comencé el Doctorado (de ahí que su nombre sean las siglas del Departamento). No leerán ni esta Tesis ni estas líneas, pero me ha parecido justo poner de manifiesto todo el cariño que ellos me han regalado todo este tiempo y de forma desinteresada (aunque quizás el servirles su comida favorita haya tenido que ver).

Mi agradecimiento más sincero va también para todos aquellos que me han acompañado y/o siguen haciéndolo en el día a día de mi *primera casa*, la Facultad. **Eduardo Huedo**, el temido *Doctor H*, siempre ha estado para darme esos consejos científicos y personales para sobrevivir en ciertos *flatworlds*. **Antonio Juan (Toñete) Rubio** me enseñó a hablar el idioma administrativo, que me fue muy útil cuando comenzaba el Doctorado, y a ponerle al mal tiempo, cara de indiferencia y seguir tirando. **Constantino (Tino) Vázquez**, **Javer Fontán** y **Luis González**, no sólo han

dado el apoyo técnico necesario para esta Tesis, sino que además y gracias a ellos, todas las actividades sociales congresiles nocturnas en las que hemos coincidido me han servido para evadirme del estrés y sentirme muy vivo. A **Abel**, que ha sido alumno mío y ahora compañero en el Grupo de Investigación, le tengo que agradecer el también haber estado allí, ya sea para alguna consulta técnica como para practicar Capoeira (también de lo mejor contra el estrés). De entre los que más me soportan diariamente, quiero destacar a los *moradores del despacho 347* tanto actuales como pasados, y en particular a **Guillermo Botella**, **David Sigüenza**, **Miguel Peón**, **Angel Luis González**, **Jesús Fernández**, y los recién incorporados **Juan Antonio Clemente** y **Pablo García**. Mi mayor gratitud por hacerme placentera mi estancia en esas cuatro paredes.

Aun no estando en mi Departamento, **Antonio Fuentes** es y ha sido uno de mis mejores compañeros de Grupo de Investigación. Desde aquellas Jornadas Técnicas de RedIRIS en Toledo en la que me consiguió todos los regalos de los expositores al grito de *venga, dale algo al chaval, que está comenzando en esto de la Investigación*, me ha enseñado a que cuando parece que ya no se puede dar más en el trabajo, siempre se podrá sacar el doble. Para mí es la figura del trabajador nato, así como del mejor amigo que alguien pueda desear. Por supuesto, no puedo olvidarme de su mujer, **Angelines Alberto**, a la que pido encarecidamente que le obligue a bajar el ritmo de trabajo, pues le queremos tener durante mucho tiempo.

Mucho tengo que agradecer a mi Departamento, de ahí que quiera dedicar un apartado a **Milagros Fernández** y **Paco Tirado**, directores del mismo, por su hospitalidad. A **Daniel Mozos** quiero agradecerle, no solamente que haya aceptado revisar esta Tesis, sino el hecho de haberme enseñado Perl en la Escuela Complutense de Verano de Bioinformática del 2003, y que de alguna manera, supuso una gran ayuda para mi posterior Investigación. Mi gratitud va también para los dos secretarios académicos que he conocido: **Ignacio (Iñaki) Hidalgo** y **Daniel Chaver**, y especialmente al último por toda la lata que le he dado con la tramitación de esta Tesis. Para finalizar en mi Departamento, hay dos *mujeres Dacya* a las que quiero dar mi más sincero agradecimiento: **Sara Román** y **Guadalupe Miñana**, por su inestimable ayuda cuando comencé mi andadura en la docencia.

Cambiando al Departamento de Ingeniería del Software e Inteligencia Artificial, no puedo olvidarme de **Miguel Vázquez**, **Raquel Hervás**, **Pablo Moreno** y **Virginia Francisco**. El hecho de haber comenzado juntos, en algún caso haber coincidido en algún congreso, ha hecho que estemos unidos. Más de una vez nos hemos montado nuestra propia Junta de Facultad a la hora del café o en cualquier evento social que organizáramos como mera excusa para quedar.

Pero no menos importantes, son los proyectos que permitieron expandir mi propio Grid humano, pues siempre he sostenido que *dos máquinas no pueden trabajar*

juntas si no hay una interacción humana antes. Así comienzo mencionando al proyecto EGEE (*Enabling Grids for E-SciencE*), financiado por la Comisión Europea (contrato número INFSO-RI-031688) a través del Sexto Programa Marco. Para mí ha sido y es un honor trabajar en esta infraestructura, calificada hace 1 año como una de las *7 Maravillas Mundiales de las Tecnologías de la Información*[1].

Creo justo comenzar la relación de agradecimientos relativos a EGEE con **Paco Castejón**, el *ecologista de la fusión* del CIEMAT. Para mí, ferviente lector de Isaac Asimov desde los 8 años, el haber iniciado mi andadura con una aplicación de Física de Fusión, me pareció algo de otro mundo. El tiempo que Paco me ha dedicado, enseñándome incluso su *bebé* (el TJ-II) y explicándome todo lo que sé ahora mismo de Fusión, me ha parecido el mejor de los regalos. Sin moverme del CIEMAT, quiero darle las gracias también a **Miguel Cárdenas**, por haberme servido de guía en muchos aspectos, científicos y humanos. Jamás podré olvidar como de forma desinteresada, me aconsejó sobre todo los procedimientos que debía seguir durante mi primera visita al CERN.

Mi siguiente agradecimiento es para **Christophe Blanchet**, coordinador de las aplicaciones de Biomedicina. Lo que comenzó con una animada charla sobre navegación a vela en la cantina del CERN hace 2 años, ha evolucionado a la elaboración de uno de los informes de la presente Tesis y todo un panorama de colaboraciones futuras.

También dentro del CERN conocí a **Massimo Lamanna**, al que tengo que agradecer no sólo su inmejorable hospitalidad durante mi primera visita y en los congresos posteriores que organizó, sino además el haber redactado el otro informe.

Gracias a EGEE pude conocer a **Vicente Hernández**, de la Universidad Politécnica de Valencia. Sus dotes de organización científica han permitido crear la Red Española de E-Ciencia, en la que para mí es un privilegio participar, sobre todo para aprender de su amplia experiencia.

Sería un error lamentable olvidarme del grupo del INFN de Catania, liderado por **Roberto Barbera**, al que debo agradecer su amabilidad y hospitalidad, no sólo en sus instalaciones en el CERN, sino el trato que me han dispensado siempre. En particular, la colaboración con **Antonio (Tony) Calanducci** ha desembocado en grandes proyectos de futuro.

También un proyecto del Ministerio de Educación y Ciencia, *Una Infraestructura Grid para Utility Computing* (contrato número TIN2006-02806) me permitió no solo viajar, sino compartir conocimientos y vivencias tanto científicas como personales.

Una de las personalidades que he tenido el gusto de conocer ha sido **Rajkumar Buyya**, cuyos artículos han servido de base para la presente Tesis. Nuestro primer contacto tuvo lugar durante las Jornadas de Paralelismo de Albacete, donde me de-

[1] http://www.cio.com/article/135700/Seven_Wonders_of_the_IT_World/4

mostró que el hecho de ser un gurú no está reñido con el dispensar un trato exquisito con los que empiezan. Congreso tras congreso en los que hemos coincidido, siempre ha venido a saludarme independientemente de con quién estuviera hablando en ese momento. Para mi, todo un ejemplo de calidad humana dentro de la Ciencia.

También he encontrado mi inspiración en las conferencias impartidas por **Manuel Delfino**, del PIC de Barcelona, tanto en el estrado como fuera de él. Su trayectoria profesional siempre me ha parecido envidiable y en más de una ocasión, su forma de hacer las cosas, otro ejemplo más a seguir. Desde luego, sin alejarnos de esta institución, quiero agradecer a **Kai Neuffer** por la ayuda prestada cuando comencé en EGEE.

Sin salir de Barcelona, mi agradecimiento más sincero a **Rosa Badía**, la *mamá* del MareNostrum. Su gran personalidad ha marcado de manera especial la colaboración de nuestros grupos en los diferentes proyectos.

Y finalmente otro proyecto llamado BioGridNet, de la Comunidad de Madrid (número de contrato S-0505/TIC-0101) me devolvió a mi querida Bioinformática, cuyo descubrimiento realicé durante una Escuela Complutense de Verano[2] en el año 2003. Nunca me ha dejado de cautivar este campo. También me permitió, deslumbrado por la grandeza de la infraestructura de EGEE, asistir y tomar parte activa en el nacimiento de una más pequeña pero no menos ambiciosa: GRIDIMadrid[3].

Dentro de este proyecto he tenido el privilegio de colaborar con el grupo de **Alfonso Valencia** en el CNIO. Además, quiero dar las gracias de forma encarecida a **José María Fernández**, porque fruto de su colaboración han salido (y saldrán) bastantes publicaciones. Ya fuera de noche, en fin de semana, siempre he obtenido sus respuestas acerca del funcionamiento de CD-HIT.

BioGridNet me ha permitido viajar fuera de España. De esta forma, he podido entrar en contacto con el grupo de **Domenico Laforenza** en el CNR de Pisa. A él debo muchísimo, estando indeciso entre el porcentaje de científico y personal (diría que 100 % ambos, aunque pueda resultar imposible). La acogida que me brindó en mi primera estancia, haciéndome sentir como uno más de los suyos, hizo que repitiera, amén de querer volver siempre. De igual forma, quiero dar las gracias a los integrantes de su grupo con los que he tenido un montón de vivencias profesionales y personales: **Ranieri Baraglia**, por esos consejos desde su amplia experiencia delante de una taza del mejor café del mundo; **Nicola Tonellotto**, **Patrizio Dazzi** y **Marco Pasquali**, porque siendo mis *Cicerones* acabamos siendo inmejorables amigos; **Diego Puppin**, por ayudarme a compaginar trabajo con Capoeira; **Stefania Lombardi**, por sus conversaciones sobre temas filosóficos y también su amistad.

Aquí también quiero agradecer a la que llamo cariñosamente *La Tribu OTM*, por-

[2] http://www.ucm.es/info/fgu/escuelas/verano/
[3] http://www.gridimadrid.org/

que fue gracias a uno de sus congresos federados (GADA) que entré en contacto con **María S. Pérez** y **Pilar Herrero** de la Universidad Politécnica de Madrid. Gracias a ellas fui Program Committee por primera vez en mi vida. Ellas me enseñaron a conocer el trabajo de los demás durante el congreso y, lo más imporante, a conocer a los demás después del congreso. También gracias a estos congresos conocí a **Omer F. Rana**, de la Universidad de Cardiff, que además de ser una persona de gran cultura, el trato dispensado por él es excelente.

Y aquí finalizo este capítulo de agradecimientos que creo ha sido lo suficientemente exhaustivo. Y si me he olvidado de alguien, desde luego no ha sido de forma intencional, mil perdones. A todos los mencionados, gracias, muchas gracias de corazón, **esta Tesis está dedicada a todos vosotros**.

José Luis Vázquez Poletti
Madrid
21 de junio de 2008

Acerca de este Documento

Esta Tesis Doctoral se presenta como compendio de publicaciones editadas, de acuerdo con el epígrafe 4.4 de la *Normativa de desarrollo de los artículos 11, 12, 13 y 14 del Real Decreto 56/2005, de 21 de Enero, por el que se regulan los estudios universitarios oficiales de postgrado de la Universidad Complutense* (Aprobado en Consejo de Gobierno con fecha 13 de Junio de 2005 y publicado en el BOUC con fecha de 5 de Julio de 2005).

Los artículos que se aportan como parte de la Tesis Doctoral son los siguientes:

- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *Coordinated Harnessing of the IRISGrid and EGEE Testbeds with GridWay*. Journal of Parallel and Distributed Computing, vol. 66, n. 5 (IPDPS'04 Special Issue), Mayo 2006, pp. 763-771.
- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *A Comparison Between two Grid Scheduling Philosophies: EGEE WMS and GridWay*. Multiagent and Grid Systems, vol. 3, n. 4, 2007, pp. 429-439.
- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *Execution of a Bioinformatics Application in a Joint IRISGrid/EGEE Testbed*. LaSCoG Workshop, 6th PPAM Conference, Poznan (Polonia), Septiembre 2005. Lecture Notes in Computer Science (LNCS). Vol. 3911, pp. 831-838, 2006. Springer Verlag.
- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *A Comparative Analysis between EGEE and GridWay Workload Management Systems*. International Conference on Grid computing, high-performAnce and Distributed Applications on the Move Federated Conferences (GADA) 2006, Montpellier (Francia), Noviembre 2006. Lecture Notes in Computer Science (LNCS). Vol. 4276, pp. 1143-1151, 2006. Springer Verlag.
- F. Castejón, I. Campos, A. Cappa, L.A. Fernández, E. Huedo, I.M. Llorente, V. Martín, R.S. Montero, M. Mikhailov, A. Tarancón, M.A. Tereschenko, J.L. Vázquez-Poletti, J.L. Velasco, V. Voznesensky. *Fusion in the Grid*. Spanish Conference on e-Science Grid Computing, CIEMAT Madrid (España), Marzo 2007.
- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *Advanced Strategies for Efficient Workflow Management in a Protein Clustering Application*

with GridWay. 1st Iberian Grid Infrastructure Conference, Santiago de Compostela (España), Mayo 2007. CESGA, pp. 199-207.

- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *Workflow Management in a Protein Clustering Application*. 5th International Workshop on Biomedical Computations on the Grid (BioGrid'07) on the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007), Rio de Janeiro (Brasil), Mayo 2007. IEEE Computer Society Press, pp. 679-684.

- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *Replication Heuristics for Efficient Workflow Execution on Grids*. International Conference on Grid computing, high-performAnce and Distributed Applications at on the Move Federated Conferences (GADA) 2007, Vilamoura (Portugal), Noviembre 2007. Lecture Notes in Computer Science (LNCS). Volume 4805, pp. 31-32, 2007. Springer Verlag.

- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *CD-HIT Workflow Execution on Grids using Replication Heuristics*. 1st International Workshop on Modern Computer Tools for the Biosciences - A Grid Perspective - (ModernBio08) on the 8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2008), Lyon (Francia), Mayo 2008. IEEE Computer Society Press, pp. 735-740.

Este documento, según la normativa vigente, expone una introducción al trabajo realizado, acompañada de una revisión del estado del arte del campo. También se describe el planteamiento de los objetivos del trabajo y se incluye una discusión integradora sobre el contenido de los artículos presentados. Finalmente, se incluyen las conclusiones del trabajo y una sección de referencias bibliográficas que integran y complementan las ya incluidas en los artículos que componen el núcleo de esta Tesis Doctoral.

Por otro lado, este trabajo de Tesis Doctoral opta a la obtención de la mención *Doctor Europeus* según las indicaciones del epígrafe 7 de la misma normativa. De acuerdo con los requisitos de la normativa, partes de la memoria de Tesis se deben presentar en un idioma oficial de la Unión Europea distinto del castellano. Por este motivo, el resumen y el apartado de principales aportaciones y trabajo futuro aparecen en Español, Italiano e Inglés.

Su questo Documento

Questa Tesi Dottorale si presenta come un compendio di pubblicazioni, d´accordo con l´epigrafe 4.4 della *Normativa de desarrollo de artículos 11, 12, 13 e 14 dal Real Decreto 56/2005, 21 Gennaio, che regolano gli studi della Universidad Complutense di Madrid* (Approvata nel Consiglio di Governo il 13 Giunio 2005 e pubblicato nel BOUC il 5 Luglio 2005).

Gli articoli presentati come parte della Tesi Dottorale sono:

- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *Coordinated Harnessing of the IRISGrid and EGEE Testbeds with GridWay*. Journal of Parallel and Distributed Computing, vol. 66, n. 5 (IPDPS'04 Special Issue), Maggio 2006, pp. 763-771.

- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *A Comparison Between two Grid Scheduling Philosophies: EGEE WMS and GridWay*. Multiagent and Grid Systems, vol. 3, n. 4, 2007, pp. 429-439.

- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *Execution of a Bioinformatics Application in a Joint IRISGrid/EGEE Testbed*. LaSCoG Workshop, 6th PPAM Conference, Poznan (Polonia), Settembre 2005. Lecture Notes in Computer Science (LNCS). Vol. 3911, pp. 831-838, 2006. Springer Verlag.

- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *A Comparative Analysis between EGEE and GridWay Workload Management Systems*. International Conference on Grid computing, high-performAnce and Distributed Applications on the Move Federated Conferences (GADA) 2006, Montpellier (Francia), Novembre 2006. Lecture Notes in Computer Science (LNCS). Vol. 4276, pp. 1143-1151, 2006. Springer Verlag.

- F. Castejón, I. Campos, A. Cappa, L.A. Fernández, E. Huedo, I.M. Llorente, V. Martín, R.S. Montero, M. Mikhailov, A. Tarancón, M.A. Tereschenko, J.L. Vázquez-Poletti, J.L. Velasco, V. Voznesensky. *Fusion in the Grid*. Spanish Conference on e-Science Grid Computing, CIEMAT Madrid (Spagna), Marzo 2007.

- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *Advanced Strategies for Efficient Workflow Management in a Protein Clustering Application*

with GridWay. 1st Iberian Grid Infrastructure Conference, Santiago de Compostela (Spagna), Maggio 2007. CESGA, pp. 199-207.

- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *Workflow Management in a Protein Clustering Application*. 5th International Workshop on Biomedical Computations on the Grid (BioGrid'07) on the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007), Rio de Janeiro (Brasile), Maggio 2007. IEEE Computer Society Press, pp. 679-684.

- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *Replication Heuristics for Efficient Workflow Execution on Grids*. International Conference on Grid computing, high-performAnce and Distributed Applications at on the Move Federated Conferences (GADA) 2007, Vilamoura (Portugal), Novembre 2007. Lecture Notes in Computer Science (LNCS). Volume 4805, pp. 31-32, 2007. Springer Verlag.

- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *CD-HIT Workflow Execution on Grids using Replication Heuristics*. 1st International Workshop on Modern Computer Tools for the Biosciences - A Grid Perspective - (ModernBio08) on the 8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2008), Lyon (Francia), Maggio 2008. IEEE Computer Society Press, pp. 735-740.

Questo documento, adempiendo alla normativa vigente, consta di una relazione introduttiva al lavoro realizzato, affiancata da una panoramica della situazione del campo di studio. La descrizione dell´ approccio metodologico agli obbiettivi include una discussione integrativa sul contenuto degli articoli presentati. La parte dedicata alle conclusioni seguita da una sezione di referenze bibliografiche che integrano quelle gi citate negli articoli che compongono questa Tesi dottorale.

Dal momento che questa Tesi Dottorale opta alla menzione *Doctor Europeus*, come indicato dall´ epigrafe 7 della normativa citata, estratti del testo devono essere presentati in una lingua ufficiale dellÚnione Europea, diversa dallo spagnolo. Per questo motivo il riassunto, la sezione delle principali contribuzioni e le previsibili prospettive future appaiono in Spagnolo, Italiano e Inglese.

About this Document

This work is presented as a binding of edited publications, according to section 4.4 of the *Normativa de desarrollo de los artículos 11, 12, 13 y 14 del Real Decreto 56/2005, de 21 de Enero, por el que se regulan los estudios universitarios oficiales de postgrado de la Universidad Complutense* (Approved by the Consejo de Gobierno as of June 13th 2005 and published in the BOUC as of July 5th 2005).

The articles presented are the following:

- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *Coordinated Harnessing of the IRISGrid and EGEE Testbeds with GridWay*. Journal of Parallel and Distributed Computing, vol. 66, n. 5 (IPDPS'04 Special Issue), May 2006, pp. 763-771.

- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *A Comparison Between two Grid Scheduling Philosophies: EGEE WMS and GridWay*. Multiagent and Grid Systems, vol. 3, n. 4, 2007, pp. 429-439.

- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *Execution of a Bioinformatics Application in a Joint IRISGrid/EGEE Testbed*. LaSCoG Workshop, 6th PPAM Conference, Poznan (Polonia), September 2005. Lecture Notes in Computer Science (LNCS). Vol. 3911, pp. 831-838, 2006. Springer Verlag.

- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *A Comparative Analysis between EGEE and GridWay Workload Management Systems*. International Conference on Grid computing, high-performAnce and Distributed Applications on the Move Federated Conferences (GADA) 2006, Montpellier (Francia), November 2006. Lecture Notes in Computer Science (LNCS). Vol. 4276, pp. 1143-1151, 2006. Springer Verlag.

- F. Castejón, I. Campos, A. Cappa, L.A. Fernández, E. Huedo, I.M. Llorente, V. Martín, R.S. Montero, M. Mikhailov, A. Tarancón, M.A. Tereschenko, J.L. Vázquez-Poletti, J.L. Velasco, V. Voznesensky. *Fusion in the Grid*. Spanish Conference on e-Science Grid Computing, CIEMAT Madrid (España), March 2007.

- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *Advanced Strategies for Efficient Workflow Management in a Protein Clustering Application*

with GridWay. 1st Iberian Grid Infrastructure Conference, Santiago de Compostela (España), Mayo 2007. CESGA, pp. 199-207.

- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *Workflow Management in a Protein Clustering Application*. 5th International Workshop on Biomedical Computations on the Grid (BioGrid'07) on the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007), Rio de Janeiro (Brasil), May 2007. IEEE Computer Society Press, pp. 679-684.

- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *Replication Heuristics for Efficient Workflow Execution on Grids*. International Conference on Grid computing, high-performAnce and Distributed Applications at on the Move Federated Conferences (GADA) 2007, Vilamoura (Portugal), November 2007. Lecture Notes in Computer Science (LNCS). Volume 4805, pp. 31-32, 2007. Springer Verlag.

- J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *CD-HIT Workflow Execution on Grids using Replication Heuristics*. 1st International Workshop on Modern Computer Tools for the Biosciences - A Grid Perspective - (ModernBio08) on the 8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2008), Lyon (Francia), May 2008. IEEE Computer Society Press, pp. 735-740.

According to the normative, this document also includes an introduction and a study of the current state of the art in the domain. There is also a description of the objectives proposed for this work and a discussion integrating the contents of the six articles and relating them to the aforementioned objectives. Finally, there is also a section devoted to analysing the results and outlining the conclusions and future work. Finally, all the references from the contributed articles are integrated and complemented in a unified bibliography.

Additionally, this PhD. thesis is presented as a candidate to the obtention of the *Doctor Europeus* mention, following the indications of section 7 from the same normative. According to the regulations, parts of this work must be presented in an oficial language of the European Union other than Spanish. For this reason, the summary and the Section containing the main contributions and future work are all presented in Spanish, Italian and English.

Resumen

Entre los diferentes aspectos que rodean a la Computación Grid, esta Tesis Doctoral se centra en la ejecución eficiente de flujos de trabajos. Los flujos de trabajos reflejan las necesidades de complejidad del ser humano, y la Computación Grid comienza a ser madura para resolver problemas científicos, puesto que permite el acceso a una gran cantidad de recursos. No obstante, debido a su naturaleza inherente, el Grid no está completamente listo para ejecutar bastantes tipos de flujos de trabajos en un tiempo razonable. Con el objetivo de justificar la adopción de este paradigma de la computación, se deben analizar los algoritmos que rigen estos flujos de trabajos y de ahí, implantar optimizaciones para que su ejecución gane eficiencia.

Este trabajo comienza con una introducción a la Computación Grid donde se explican sus principales componentes y se describen las infraestructuras más significativas. Como los flujos de trabajos son implantados por aplicaciones, este Capítulo servirá también de introducción al portado, por lo que se hablará de diferentes tecnologías a diferentes niveles.

El tipo más sencillo de flujo de trabajos, las aplicaciones de alta productividad, son las primeras en ser estudiadas. Así, una aplicación de Física de Fusión suministrada por el CIEMAT[4] necesitaba incrementar la cantidad de datos procesados. La Computación Grid garantizó al principio la eficiencia necesaria. Cuando esta cantidad aumentó nuevamente, se adoptó una estrategia del tipo *chunk* con buenos resultados.

La complejidad se incrementó en el Capítulo siguiente, donde una aplicación bioinformática propuesta por el CNIO[5] encuentra en el Grid la mejor solución para evitar las restricciones de memoria de una única máquina. Como la ejecución en el Grid no estaba ahorrando suficiente tiempo, se estudiaron diferentes heurísticas de optimización. De éstas, se eligieron dos para su implantación y estudio posterior. Adicionalmente, este trabajo presenta un modelo válido para predecir el tiempo de ejecución según las condiciones de partida, también usando las heurísticas implantadas.

Después de un resumen de las principales aportaciones y trabajo futuro, este documento finaliza con un Apéndice. Este Apéndice contiene las publicaciones men-

[4] http://www.ciemat.es/
[5] http://www.cnio.es/

cionadas en el apartado *Acerca de este Documento*.

Riepilogo

Questa Tesi Dottorale è centrata, di tutti gli aspetti che riguardono la Computazione Grid, sull´ esecuzione efficiente di workflows. I workflows rispecchiano la necessità di complessit dell´ essere umano e la Computazione Grid è ormai matura per risolvere problemi scientifici in quanto permette l´ accesso a una ingente quantità di risorse. Attualmente il Grid non è ancora in grado di processare certi tipi di workflows in tempi accettabili. Per giustificare l´ adozione di questo paradigma della computazione, bisogna analizzare gli algoritmi che definiscono questi workflows cosí da poter, in un secondo tempo, implementare ottimizzazioni che rendano più efficiente la loro esecuzione.

Questo lavoro inizia con un´ introduzione alla Computazione Grid in cui si analizzano le componenti principali e si descrivono le infrastrutture più significative. Dal momento che i workflows sono implementati da applicazioni, questo Capitolo servirà anche da introduzione al porting, in quanto si parlerà di diverse tecnologie a diversi livelli. Il tipo più semplice di workflows, le applicazioni di alta produttività, sono le prime ad essere studiate. Così si prende in esame un´ applicazione di Fisica di Fusione somministrata dal CIEMAT[6] che aveva bisogno di incrementare la quantità dei dati processati. La Computazione Grid garantì l´ efficienza necessaria allínizio. Quando la quantità dei dati fu nuovamente incrementata, si adottò una strategia del tipo *chunk*, ottenendo buoni risultati.

La complessità si vide incrementata nel Capitolo seguente, quando l´ applicazione bioinformatica proposta dal CNIO trovò nel Grid la miglior soluzione alle restrizioni di memoria di una singola macchina. Dal momento che l´ esecuzione nel Grid non presentava un significativo risparmio di tempo, si studiarono diverse euristiche di ottimizzazione. Se ne scelsero due per l´ implementazione e posteriore studio. Questo lavoro presenta inoltre un modello valido per predire il tempo di esecuzione secondo le condizioni di partenza, usando anche le euristiche implementate.

Dopo un riepilogo delle principali contribuzioni e sviluppi futuri, questo documento si conclude con un´ Appendice dove si elencano le pubblicazioni menzionate nella sezione *Su questo Documento*.

[6] http://www.ciemat.es/

Summary

Among the different aspects that involve Grid Computing, efficient execution of workflows is focused by the present work. Wokflows do mirror human needs of complexity and Grid Computing technology starts to be mature for solving scientific problems, as it allows access to a great amount of resources. Nevertheless, due to its inherent nature, the Grid is not completely ready for executing many wokflow types in a reasonable time. In order to justify the adoption of this computing paradigm, the algorithms governing these workflows must be analyzed and optimizations must be implemented so their execution will gain efficiency.

This work starts with an introduction to Grid Computing where its main components are explained and some significant infrastructures are described. As workflows are implemented by applications, this Chapter will serve also as an introduction to Grid porting, reviewing existing technologies at different levels.

The simplest type of workflow, the high throughput applications, is studied at first. Here, a Fusion Physics application provided by CIEMAT[7] needed to increase its amount of processed data. At the beginning, Grid Computing provided the needed efficiency. When this data need increased again, a *chunk* strategy was adopted obtaining good results.

Complexity is increased at the following Chapter, where a bioinformatic application proposed by CNIO[8] finds in the Grid the best solution to bypass the memory restrictions of a single machine. As the Grid execution of this application was not saving enough time, many optimization heuristics weres studied. From these, two have been chosen for implementation and further study. Additionally, this work provided a valid model to predict the execution time for a given set of starting conditions, also using the implemented heuristics.

After a summary of main contributions and future work, this document ends with an Appendix. This Appendix contains the publications mentioned by the *About this Document* notice.

[7] http://www.ciemat.es/
[8] http://www.cnio.es/

Capítulo 1

Introducción

> I do not fear computers. I fear the lack of them.
>
> Isaac Asimov

Este capítulo, primero de la presente Tesis Doctoral, tiene por objetivo ofrecer una visión general del entorno sobre el que se basa la Investigación realizada. De esta forma, se introducirá a continuación la Computación Grid, para luego hablar de las principales infraestructuras Grid existentes como ejemplos característicos en la Sección 1.2. Posteriormente, las tecnologías Grid serán introducidas en la Sección 1.3, y el portado de aplicaciones, en la Sección 1.4. A lo que el portado se refiere, se describirán las soluciones de programación existentes en la Sección 1.5, así como portales y compositores de servicios en la Sección 1.6.

1.1. Computación Grid

Computación Grid es un término que hace referencia a un nuevo paradigma de la Computación que basa su infraestructura en Internet. Su fin es compartir, a gran escala y de forma desacoplada, recursos distribuidos geográficamente. El objetivo de una infraestructura Grid no es únicamente el constituir *El Ordenador más grande del Mundo* a fuerza de unir ordenadores más pequeños, sino también proveer mecanismos que faciliten tanto la compartición como el acceso de sus recursos. La idóneidad del Grid radica en la evolución de las redes de comunicación de alta velocidad, haciendo que las funcionalidades ofrecidas por la tecnología Grid guarden un paralelismo con el servicio de suministro eléctrico (ubicuidad, transparencia, acceso).

La Computación Grid ha sido el resultado de la evolución de otros paradigmas. Cuando, a mediados de los años 90, se hicieron populares ciertas tendencias de

computación en red (*Net Computing*), la hegemonía centralizadora (servicios ofrecidos por un solo sistema) en la computación de altas prestaciones tocó a su fin. Estas tendencias son:

- **Cluster Computing:** La computación es realizada por equipos con la misma arquitectura unidos por redes de interconexión de 100 o 1000 MBits/segundo. El control de los trabajos enviados es centralizado al realizarse desde una máquina determinada y el paradigma de programación empleado es el de memoria distribuida. Como ejemplos de sistemas de gestión *Cluster* tenemos MOSIX[1] y OpenPBS[2].

- **Intranet Computing:** Se trata de una mejora del *Cluster Computing*, cuya limitación radicaba en la homogeneidad en la arquitectura de los equipos. Este paradigma asume que los recursos de procesamiento pertenecen a la red interna de una empresa, por lo que pueden estar mucho más distribuidos y además no estar dedicados. Algunos sistemas de gestión pertenecientes a este paradigma son SGE[3] y Condor[4].

- **Internet Computing:** Este paradigma pretende ampliar la escala de la computación, allá donde los anteriores no llegan. En la mayoría de sus implantaciones, los equipos destinados al proceso pertenecen a un programa de voluntariado, basándose en el subparadigma *Public Resource Computing*, como por ejemplo Seti@Home[5]. Existe otro subparadigma denominado *On-Demand Computing* en el cual, las aplicaciones emplean recursos específicos de forma remota, como es el caso de Netsolve[6].

- **Peer-to-peer Computing:** Al contrario que en las soluciones antes expuestas, basadas en un modelo cliente/servidor, este paradigma considera la existencia de comunicación entre todos los nodos de trabajo. Esta concepción reniega de la centralización salvo en el caso de los servidores de información. Implantaciones de este paradigma son las aplicaciones de intercambio de ficheros por Internet, de sobra conocidas por el usuario de informática básica (E-Mule[7], Gnutella[8]).

[1] http://www.mosix.org/
[2] http://www.openpbs.org/
[3] http://www.sun.com/software/gridware/
[4] http://www.cs.wisc.edu/condor/
[5] http://setiathome.berkeley.edu/
[6] http://icl.cs.utk.edu/netsolve/
[7] http://www.emule-project.net/
[8] http://www.gnutella.com/

- **Collaborative & Computing Portals:** Focalizándose en el usuario, este paradigma pretende ofrecer un acceso sencillo y eficiente (empleando el navegador web) a los servicios de computación, así como a instrumentos de colaboración entre las entidades participantes. Un ejemplo de estos sistemas es ExPASy[9].

1.2. Infraestructuras Grid

Analizando más en detalle los paradigmas anteriores en los que el Grid obtiene sus características, observamos que *Cluster* e *Intranet Computing* ofrecen al usuario una gran capacidad de procesamiento, pero dentro de un único dominio de administración. Un paso más allá lo realizan herramientas como SGE Enterprise o Condor Flocking [ELvD+96] que, a cambio de emplear la misma herramienta de gestión y sacrificar la heterogeneidad en políticas de seguridad y administración de recursos, permiten la interconexión de varios departamentos o dominios de administración. Es así como la Computación Grid supone un cambio radical y una alternativa en la colaboración de sistemas conectados a Internet. Además, su enorme potencial respecto al intercambio y gestión de recursos ofrece un nuevo panorama a la Computación de Altas Prestaciones.

Volviendo a la definición romántica del *Ordenador más grande del Mundo*, podemos vislumbrar las diferentes funcionalidades que ofrece un Grid y sus elementos constituyentes. De esta forma, un Grid necesitará:

- un procesador para ejecutar los trabajos enviados, constituido por elementos de computación. Dichos elementos pueden ser casi cualquier máquina, desde un sencillo ordenador de sobremesa, hasta un complejo *cluster* de un centro de supercomputación, dotando a la infraestructura de una gran heterogeneidad;
- un disco duro para almacenar los datos de los usuarios, tanto de entrada como de salida. Los elementos que proveen esta funcionalidad también pueden ser muy variados, desde el disco duro de un simple equipo hasta un robot de cintas;
- unos mecanismos de transferencia de datos que permitan moverlos de un recurso a otro según sea necesario.
- un sistema de información que permita saber de qué recursos se dispone y qué uso se está haciendo de los mismos;
- un planificador que asigne trabajos a recursos y optimice de esta forma la carga del Grid;

[9] http://www.expasy.org/tools/

- el equivalente a los periféricos de entrada y salida, un punto de acceso que permita al usuario utilizar todas las funcionalidades del Grid;

distinguiendo a todos los elementos anteriores, el hecho de estar distribuidos y ser heterogéneos.

A continuación vamos a describir tres infraestructuras constituidas con diferentes fines, que han sido empleadas en la obtención de resultados experimentales de la presente Tesis, como ejemplos representativos de como la Computación Grid puede responder a diferentes necesidades y filosofías.

1.2.1. DSAGrid

El Grupo de Arquitectura de Sistemas de Distribuidos[10] de la Universidad Complutense de Madrid[11] dispone de una infraestructura sencilla, compuesta por pocos equipos no dedicados en exclusividad a la Computación Grid. Dadas las líneas de Investigación Grid del grupo, la infraestructura está dedicada al desarrollo, sólo ofrece servicios de ejecución y se caracteriza por su dinamismo a lo que disponibilidad de recursos y topología se refiere. Los componentes instalados, concepto del cual se hablará en la Sección 1.3, están en su última versión. Los usuarios utilizan los servicios de la infraestructura desde máquinas pertenencientes a ella y su coordinación es directa, siendo ésta otra de las razones de sus dinamismo.

1.2.2. GRIDIMadrid

El Grid de la Comunidad de Madrid[12] nace de la unión de infraestructuras como la descrita anteriormente y está dedicada a la Investigación. Cada centro participante mantiene la potestad sobre sus propios recursos pero, en el marco de esta colaboración, tiene que habilitar el acceso a los mismos para usuarios pertenecientes al resto de centros. El beneficio obtenido es el de disponer de una infraestructura mayor a cambio de una coordinación ligeramente más estrecha y compartimentalizada (Figura 1.1) definida por políticas básicas.

1.2.3. EGEE

Aumentando la escala con la que analizamos las infraestructuras Grid, llegamos a la perteneciente al proyecto EGEE (*Enabling Grids for E-sciencE*)[13]. Esta infraestructura Paneuropea de producción, cuyos datos numéricos más representativos se

[10] http://dsa-research.org/
[11] http://www.ucm.es/
[12] http://www.gridimadrid.org/
[13] http://www.eu-egee.org/

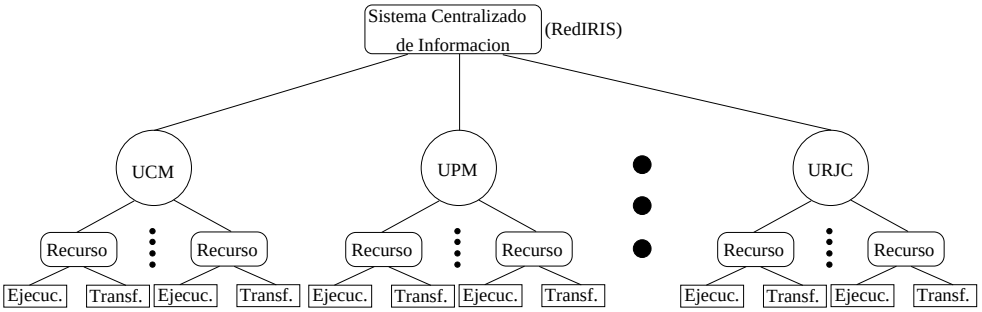


Figure 1.1: *Jerarquía del Sistema de Información y recursos de GRIDIMadrid.*

recogen en la Tabla 1.1, aglutina 240 instituciones participantes de un total de 45 países con el objetivo de ofrecer sus recursos las 24 horas del día y los 7 días de la semana. El compromiso de EGEE como Grid de producción, se extiende al uso de cada procesador, garantizando un 100 % de dedicación.

Table 1.1: *Características de la infraestructura Grid del proyecto EGEE a fecha de Marzo de 2008.*

Número de Procesadores	41mil
Almacenamiento	5 Petabytes
Trabajos concurrentes (pico)	100mil

Conviene aclarar que no todos los usuarios de esta infraestructura tienen acceso a todos sus recursos, sino que estos se reparten en las denominadas Organizaciones Virtuales. Una Organización Virtual no es más que un conjunto de recursos dedicados a un área de Investigación [FKT01], como bien puede ser Física de Fusión[14] o Biomedicina[15]. Así como un usuario sólo puede emplear los recursos destinados a su Organización Virtual, un mismo recurso puede pertenecer a diferentes Organizaciones Virtuales. En una infraestructura de estas proporciones, las políticas son muy férreas, con procedimientos definidos y regulados hasta el más mínimo detalle.

[14]http://grid.bifi.unizar.es/egee/fusion-vo/
[15]http://egee-na4.ct.infn.it/biomed/

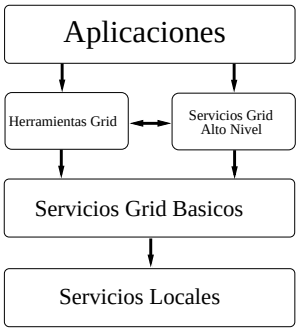


Figure 1.2: *Modelo de Capas Grid.*

1.2.4. Otras Infraestructuras Grid

Además de las infraestructuras antes descritas, existen muchas otras que no han sido empleadas en los experimentos pertenecientes a esta Investigación. Así, el proyecto EGEE ha propiciado el despliegue de infraestructuras hermanas, empleando los mismo componentes, en diferentes puntos del globo. Ejemplos de estas infraestructuras son: EELA[16] en Latinoamérica, EUMed-Grid[17] en los países del Mediterráneo, y EUChinaGrid[18] en China.

No obstante, en Europa existen otras infraestructuras independientes de la de EGEE, como es el caso de NorduGrid[19]. En Estados Unidos conviven, como ejemplos más representativos, TeraGrid[20], con una capacidad de almacenamiento total de 6 veces mayor que la de EGEE; Open Science Grid[21], con 87 instituciones participantes; y la recientemente creada TIGRE[22] que une instituciones del Estado de Texas, geográficamente dispersas y muy distantes entre sí.

1.3. Tecnologías Grid

Con el objetivo de analizar y entender la tecnología Grid, es necesario recurrir a un modelo de capas como el representado en la Figura 1.2. Se entiende como servicios locales aquellos en los recursos ofrecidos por las diferentes organizaciones

[16] http://www.eu-eela.org/
[17] http://www.eumedgrid.org/
[18] http://www.euchinagrid.org/
[19] http://www.nordugrid.org/
[20] http://www.teragrid.org/
[21] http://www.opensciencegrid.org/
[22] http://www.hipcat.net/Projects/tigre/

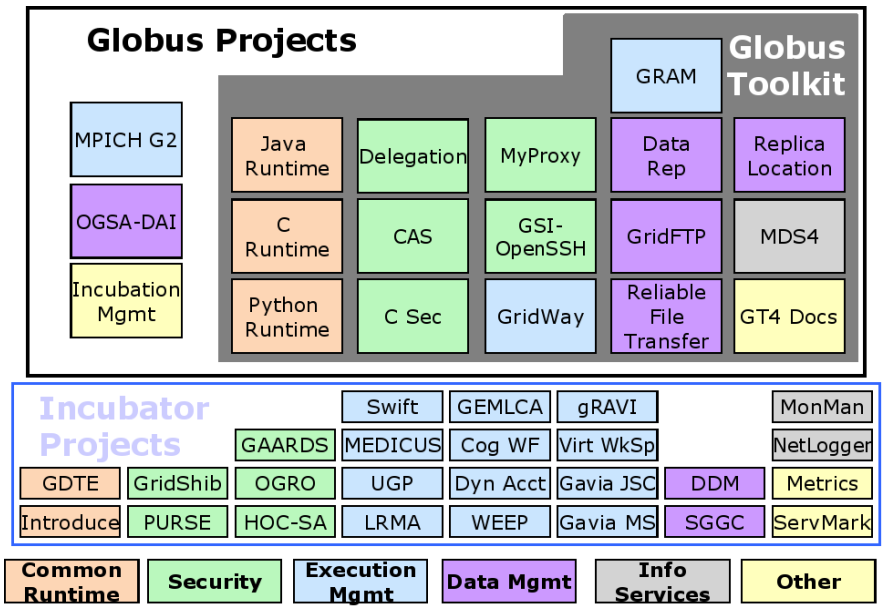


Figure 1.3: *Componentes de Globus Toolkit.*

que participan en una infraestructura Grid. Así como se había definido un recurso Grid como cualquier elemento de computación, infraestructuras del tipo de EGEE (véase la subsección 1.2.3) son más restrictivas y exigen que se trate de un sistema gestor de colas perteneciente a un *cluster*.

A continuación se encuentra la capa de servicios Grid básicos. Su misión principal es la de crear un interfaz uniforme sobre los recursos locales, independientemente de sus características. Se puede afirmar que este nivel es el fundamental del Grid, desde el cual parten el resto de servicios y herramientas. Estos servicios, construidos mediante componentes de *middleware* básico, son los que posibilitan operaciones básicas con los recursos, tales como transferir archivos, ejecutar trabajos o monitorizarlos. Actualmente, Globus Toolkit[23] se ha erigido como estándar de facto, siendo éste una colección de componentes software de arquitectura y código abiertos, diseñados para soportar el desarrollo de aplicaciones de alto rendimiento sobre entornos Grid. Estos componentes, recogidos en la Figura 1.3 son los que proporcionan un servicio básico de autenticación de usuarios, asignación de recursos, acceso remoto de datos y monitorización de trabajos.

[23]http://www.globus.org/toolkit/

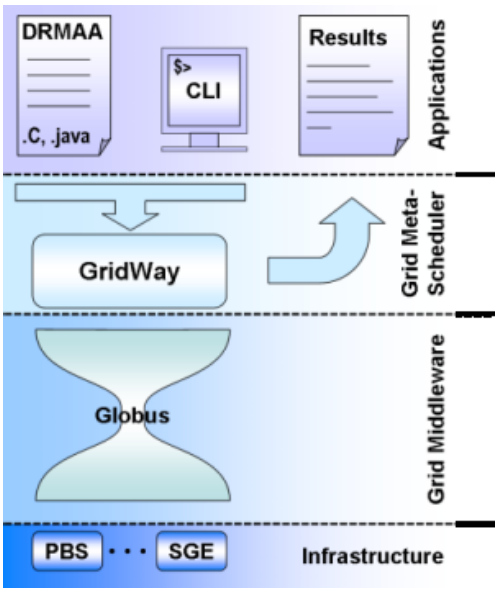


Figure 1.4: *Arquitectura de GridWay.*

No obstante, es necesaria otra capa por encima de de los servicios Grid básicos con el objetivo de que el Grid sea sencillo y eficiente de explotar. Es por ello que existen servicios y herramientas de alto nivel, dependiendo muchos del ámbito de la aplicación. Es en este nivel en el que se sitúa la función de metaplanificación. Un metaplanificador es aquel *middleware* Grid que descubre, evalúa y asigna recursos a trabajos mediante la coordinación de actividades entre los diferentes planificadores a nivel local existentes [YB05].

Existen bastantes metaplanificadores, tales como Condor/G[24] y GridBus[25], pero el empleado en esta Tesis es Globus GridWay[26], cuya arquitectura está representada en la Figura 1.4. GridWay permite una compartición eficiente de recursos de computación administrados por diferentes gestores de recursos locales, tanto de una única organización como aquellos pertenecientes a diferentes dominios de administración. Al usuario, este metaplanificador ofrece una forma sencilla y transparente de enviar trabajos al Grid, incluyendo mecanismos de planificación eficiente y tolerancia a fallos.

[24] http://www.cs.wisc.edu/condor/condorg/
[25] http://www.gridbus.org/
[26] http://www.gridway.org/

Adicionalmente, existen tecnologías Grid que ofrecen al usuario una solución integral, desde la máquina cliente hasta el recurso remoto. Una de estas tecnologías es gLite[27], producido por el proyecto EGEE. Esta tecnología hereda componentes de otros proyectos como LCG[28] y VDT[29]. El modelo de distribución consiste en construir diferentes servicios alojados en nodos diferentes, con el fin de garatizar una fácil instalación y configuración.

El Advanced Resource Connector[30] (ARC) es una solución de código abierto (licencia GPL) desarrollado en el ámbito del proyecto NorduGrid. Durante su desarrollo se enfatizó en la escalabilidad, estabilidad, fiabilidad, con el fin de facilitar la constitución de Grids de producción con un alto nivel de calidad.

El nacimiento de UNICORE[31] se debió a la necesidad de posibilitar el acceso a los recursos pertenecientes a los centros de supercomputación alemanes, heterogeneos en su conjunto. Los parámetros que guían su desarrollo son la abstracción de los servicios, la seguridad de las comunicaciones, la autonomía de cada sitio y la facilidad de uso.

La infraestructura diseñada por GRIA[32] está orientada a servicios. Su objetivo es facilitar las colaboraciones B2B (*business to business*) entre extremos organizacionales de una forma interoperable y flexible, negociando y valorando las condiciones de acceso a la vez que se focaliza en los procesos de negocios y las semánticas asociadas. Esta tecnología es gratuita y de código abierto, estando muchos de sus componentes bajo la licencia LGPL.

1.4. Introducción al Portado de Aplicaciones

Difícilmente podría entenderse un Grid sin usuarios que lo emplearan para ejecutar sus aplicaciones. Para ello, dichas aplicaciones tienen que ser portadas. El sistema resultado de este proceso es concebido como un envoltorio de la aplicación original, que le permite disfrutar de las funcionalidades de la tecnología Grid pero sin alterar las suyas propias. En el mejor de los casos, este envoltorio aislará a la aplicación (ésta no requerirá modificaciones) y convertirá al Grid en una capa transparente para la misma.

A continuación se procederá a enumerar las diferentes fases del portado de aplicaciones. Para comenzar, conviene tener en cuenta que este proceso implica un es-

[27] http://glite.web.cern.ch/
[28] http://lcg.web.cern.ch/
[29] http://vdt.cs.wisc.edu/
[30] http://www.nordugrid.org/middleware/
[31] http://www.unicore.eu/
[32] http://www.gria.org/

fuerzo inicial para conocer la aplicación objetivo. De esta forma, es importante identificar:

- los ficheros de entrada y salida, puesto que supondrán un uso de los protocolos Grid de transferencia de archivos y recursos de almacenamiento remotos;
- el perfil de la aplicación, a fin de extraer el mayor nivel de paralelismo posible y por ende, un mayor aprovechamiento del Grid;
- duración estimada de cada trabajo, siempre teniendo en cuenta que existen recursos que no aceptarán tiempos de ejecución superiores a un valor estipulado;

pero incluso antes de plantearse estos aspectos, cabe preguntarse por la viabilidad del propio portado de la aplicación. Puede suceder que la aplicación ya fuera portada a un *cluster* anteriormente, pero su salto al Grid es difícil o imposible porque, a pesar de que el Grid soporte los mismos interfaces, la heterogeneidad y dinamismo de sus recursos suelen estar reñidos con el rendimiento. Un ejemplo son algunas aplicaciones que trabajan con MPI[33].

A continuación vendrá la elección de una herramienta de alto nivel que coordine el uso de los recursos Grid en aras de una ejecución eficiente de la aplicación. Aparte de los beneficios que aporte la solución considerada, también es determinante en esta decisión el *middleware* Grid básico instalado en la infraestructura que se empleará. En los Capítulos 2 y 3 de la presente Tesis se justificará la elección de la tecnología, siempre en base a la aplicación.

Una vez realizados los pasos anteriores, será el momento de trabajar sobre un prototipo que se convertirá en una prueba de concepto. A fin de agilizar el proceso de portado, el prototipo empleará el conjunto de comandos perteneciente a la herramienta Grid de alto nivel elegida. Un sistema que emplee dichos comandos, exclusivamente orientados al Grid, no puede considerarse un sistema final en el que esta capa sea transparente al usuario, y por tanto el portado de la aplicación no podría darse por concluida.

Una vez que el prototipo obtenga un resultado exitoso, se procederá al último paso del desarrollo, en el que el usuario final es analizado. Si el usuario al cual está destinada la herramienta puede trabajar con una línea de comandos de un sistema UNIX, y por lo tanto no requiere un interfaz elaborado, se deberá optar por un API (*Aplication Programming Interface*) Grid. Obviamente, los ejecutables resultantes pueden rodearse de interfaces gráficos, como GTK[34] o TCL/TK[35], en una fase

[33] http://www.mpi-forum.org/
[34] http://www.gtk.org/
[35] http://www.tcl.tk/

posterior. Esta opción será explicada más en profundidad en la siguiente Sección.

Pero no todos los sistemas finales tienen como destino usuarios con grandes conocimientos de Informática. Es para ellos que se han desarrollado otras tecnologías complementarias. Una es la de los portales Grid, hereda características de los *Computing Portals* (véase la Sección 1.1). Otra es la de las herramientas de composición (o compositores) de servicios Grid. Ambas tecnologías serán analizadas con mayor detalle en la Sección 1.6.

1.5. Programación de Aplicaciones

Disponer de usuarios finales familiarizados con un interfaz de comandos y que la aplicación así lo permita, supone una mayor velocidad de desarrollo y optimización. En esta línea, habilitar una aplicación para ejecutarse en el Grid se consigue empleando un API ofrecida por la herramienta de alto nivel, que permite un acceso directo a sus funcionalides. En la actualidad existen bastantes API Grid y esto puede hacer que la elección se vuelva ardua. Para facilitar dicha elección, se debería tener en cuenta el hecho de que el API considerado forme parte de un estándar. Si es así, se garantizará la independencia de un solo proveedor y de ahí, la continuidad de desarrollo del propio API.

Es así que se presenta El Open Grid Forum[36] (OGF), una organización que regula los estándares en el área de la Computación Grid. Su misión consiste en ayudar a la comunidad internacional a acelerar la adopción del Grid, mediante la creación de grupos de trabajo dedicados a diferentes aspectos y disciplinas. Con esto se pretende incrementar tanto el valor comercial del Grid como ampliar los horizontes científicos que esta tecnología ofrece. El concepto de *foro* para reunir desarrolladores y usuarios de la computación distribuida (y por extensión, del Grid) comenzó a emplearse en 1998, durante una sesión del SCXX Supercomputing Conference[37]. De esta forma, la primera reunión oficial tuvo lugar en NASA Ames un año después. No fue hasta el año 2000 que el Global Grid Forum, nombre que ostentaba entonces, comenzó a celebrarse en Europa y Asia. Finalmente, tras su fusión con un grupo industrial llamado Enterprise Grid Alliance (EGA), su nombre mutó al con el que es conocido actualmente en 2004, durante su decimoctava edición. Volviendo a los estándares mantenidos por el OGF, existen varios de programación de aplicaciones, cada uno con características y objetivos diversos, que veremos a continuación.

En primer lugar se puede encontrar Simple API for Grid Applications [GJK+06] (SAGA). Esta especificación se presenta no como un reemplazo de los componentes

[36] http://www.ogf.org/
[37] http://www.supercomputing.org/

Table 1.2: *Ejemplo de programación empleando SAGA (C++).*

```
saga::job_description jd;
saga::job_service     js ("gram://remote.host.net");
saga::job             j = js.create_job (jd);
j.run ();
cout << "Job␣State:␣" << j.get_state () << endl;
j.wait ();
cout << "Retval␣" << j.get_attribute ("ExitCode") << endl;
```

middleware Grid básico como Globus, sino como un punto de acceso a dicha tecnología para desarrolladores sin demasiados conocimientos y poco tiempo para aprender. Los aspectos no funcionales que cubre son: seguridad y manejo de sesiones, manejo de permisos, operaciones y notificaciones asíncronas, monitorización, manejo de atributos, y manejo del *buffer* de entrada/salida. Por otro lado, las siguiente áreas funcionales son cubiertas por SAGA: envío y manejo de trabajos (véase el código de ejemplo en la Tabla 1.2), manejo de *namespaces*, entrada/salida de ficheros, manejo de réplicas, *streaming*, y *Remote Procedure Calls* (RPC). Finalmente, las extensiones de este API deberían cubrir los siguientes aspectos: descubrimiento de servicios, intercambio de mensajes, almacenamiento a nivel de aplicación, acceso e integración con base de datos, y manejo de *checkpoint* y recuperación. Este estándar tiene implementaciones en C++ [KMH+06] y Java [vNMW+05], ambas ofrecidas por DEISA[38].

Table 1.3: *Ejemplo de programación empleando GridRPC (cliente en C).*

```
for (i = 0; i < NUM_HOSTS; i++)
   grpc_function_handle_init(&handles[i], hosts[i],
   port, "pi/pi_trial");
for (i = 0; i < NUM_HOSTS; i++)
   gprc_call_async(&handles[i], i, times, &count[i]);
grpc_wait_all();
```

Otro estándar es GridRPC [SNM+02], que divide sus funcionalidades en dos

[38] http://deisa-jra7.forge.nesc.ac.uk/

API con destinatarios muy diferenciados: los desarrolladores de aplicaciones para el usuario final y los de *middleware*. Por un lado, el API para el usuario final permitiría el acceso a una gran cantidad de aplicaciones y sería sencillo de definir (véase un ejemplo en la Tabla 1.3). Por el otro, el API para *middleware* permitiría el desarrollo de herramientas potentes pero a cambio de una mayor complejidad.

Table 1.4: *Ejemplo de programación empleando DRMAA (Java).*

```
session.init(null);
System.out.println("Session␣Init␣success");
JobTemplate jt = session.createJobTemplate();
jt.setWorkingDirectory
        (java.lang.System.getProperty("user.dir"));
String      name;
jt.setJobName("example");
jt.setRemoteCommand("/bin/ls");
jt.setArgs(new String[] {"-l","-a"});
jt.setOutputPath("stdout." + SessionImpl.DRMAA_GW_JOB_ID);
jt.setErrorPath("stderr." + SessionImpl.DRMAA_GW_JOB_ID);
jt.setJobSubmissionState(JobTemplate.HOLD);
String id = session.runJob(jt);
System.out.println("Job␣successfully␣submitted␣ID:␣" + id);
System.out.println("Releasing␣the␣Job");
session.control(id, Session.DRMAA_CONTROL_RELEASE);
printJobStatus(id, session);
System.out.println("Synchronizing␣with␣job...");
session.synchronize(Collections.singletonList
        (Session.DRMAA_JOB_IDS_SESSION_ALL),
        Session.DRMAA_TIMEOUT_WAIT_FOREVER, false);
System.out.println("Killing␣the␣Job");
session.control(id, Session.DRMAA_CONTROL_TERMINATE);
System.out.println("The␣job␣has␣been␣deleted...");
session.deleteJobTemplate(jt);
session.exit();
System.out.println("Session␣Exit␣success");
```

El último estándar de programación de aplicaciones Grid que se abordará en esta

Tesis es el Distributed Resource Management Application API [RBC+03] (DRMAA), que recibió el estado de *full recommendation* por parte del Open Grid Forum en 2007. El objetivo de este API es cubrir todas la funcionalidades de alto nivel requeridas por las aplicaciones Grid para enviar, controlar y monitorizar trabajos en gestores locales de recursos. Entre las características que distinguen a este estándar, se encuentra:

- La cantidad de lenguajes de programación en los que está implementado. De esta forma, se puede emplear DRMAA con C/C++, Java (véase un ejemplo en la Tabla 1.4), Perl, Python y Ruby, garantizando un mayor acceso por parte de la comunidad de desarrolladores.
- El gran número de soluciones que implementan DRMAA, encontrándose entre ellos Sun Grid Engine [Gee03], Condor [CGE+05], Torque/PBS, Load Sharing Facility[39], UNICORE [RMSB06] y GridWay [HMHL04].

El hecho reflejado en el segundo punto hace de DRMAA un estándar muy interesante para el ciclo de desarrollo de cualquier aplicación Grid. En un posible escenario, el desarrollador puede contar con un *cluster* de pocas máquinas con SGE o PBS instalado, sobre las cuales crear un sistema con todas las funcionalidades finales. Una vez que el sistema está validado, éste puede llevarse al Grid cambiando la herramienta de alto nivel pero sin cambiar una sola línea de código.

1.6. Compositores de Servicios Grid y Portales Grid

Aunque se describen en esta misma Sección por encontrarse en el mismo nivel, los compositores de servicios y los portales Grid tienen características y destinatarios bien distintos. Los portales Grid, por su parte, están destinados a usuarios con pocos conocimientos de Informática, pero no es ésta su única motivación. Un portal facilita enormemente el uso a los servicios Grid puesto que emplea la tecnología Web [FGM+99], garante de universalidad en su adopción y acceso. Básicamente, añade esa capa de abstracción que exime a los usuarios de emplear directamente una máquina con *middleware* Grid instalado, pues bastará con un navegador Web desde su ubicación. Un portal Grid encapsula un sistema mayor que puede emplear scripts con comandos pertenecientes a una herramienta de alto nivel, o bien, programas creados mediante un API Grid (por ejemplo, DRMAA Perl invocado desde CGI [RC04]). Tanto en una solución como la otra deben existir sistemas ya maduros y validados, evitando que cualquier situación inesperada de fallo (no cubierta por la aplicación) pueda propagarse al portal. Es por ello que el desarrollo de un portal, de ser requerido, sea siempre el último paso en el portado de aplicaciones al Grid.

[39] http://www.platform.com/Products/platform-lsf-family/

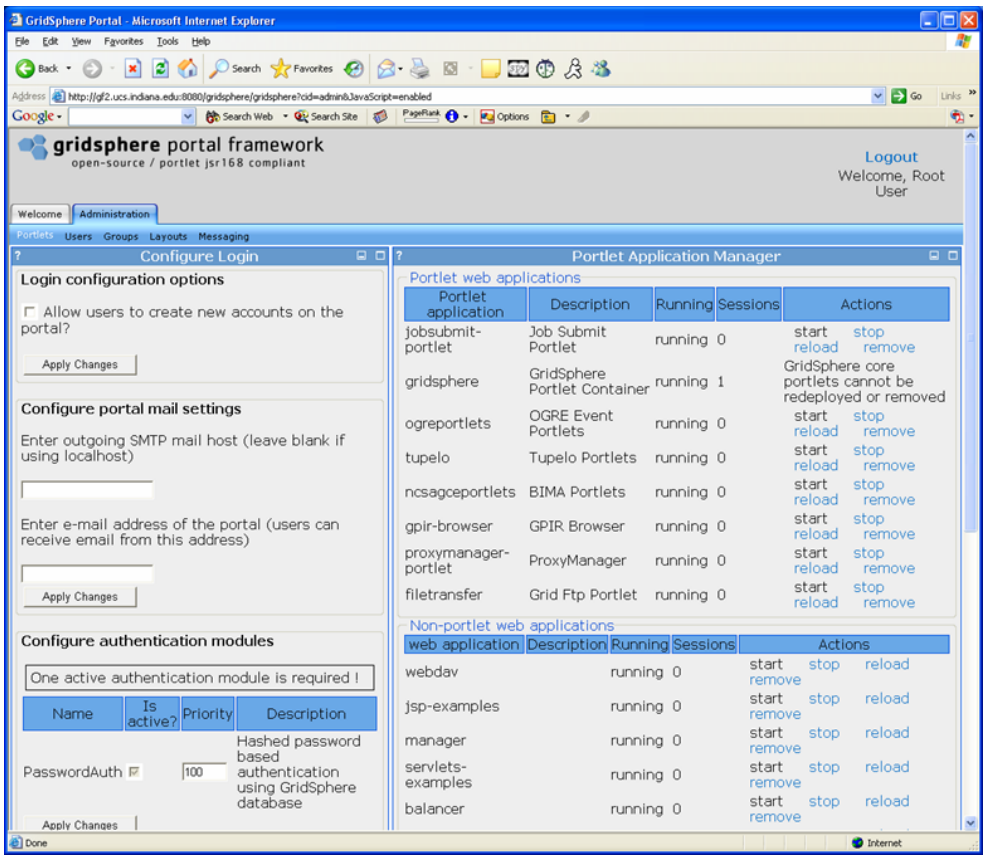


Figure 1.5: *Gestión de Portlets en GridSphere.*

En vez de construir un portal Grid desde cero, el desarrollador puede recurrir a soluciones prefabricadas como GridSphere [NRW04]. Gridsphere es un generador de portales que permite desarrollar sencillos componentes llamados *portlets*, que a su vez acceden a funcionalidades Grid. GridSphere suministra los mecanismos de autenticación y gestión de usuarios del portal, así como de las aplicaciones y funcionalidades Grid accedidas (véase la Figura 1.5). Si bien ésta es una solución muy genérica, ha servido de punto de partida para otros proyectos que han personalizado su estructura fijando unos *portlets* específicos, o directamente han discretizado muchas de sus funcionalidades por no requerir tanta potencia. Ejemplos de estos proyectos son P-GRADE [KS05], CHRONOS [CBL+05] y Telescience Portal [PLL+03].

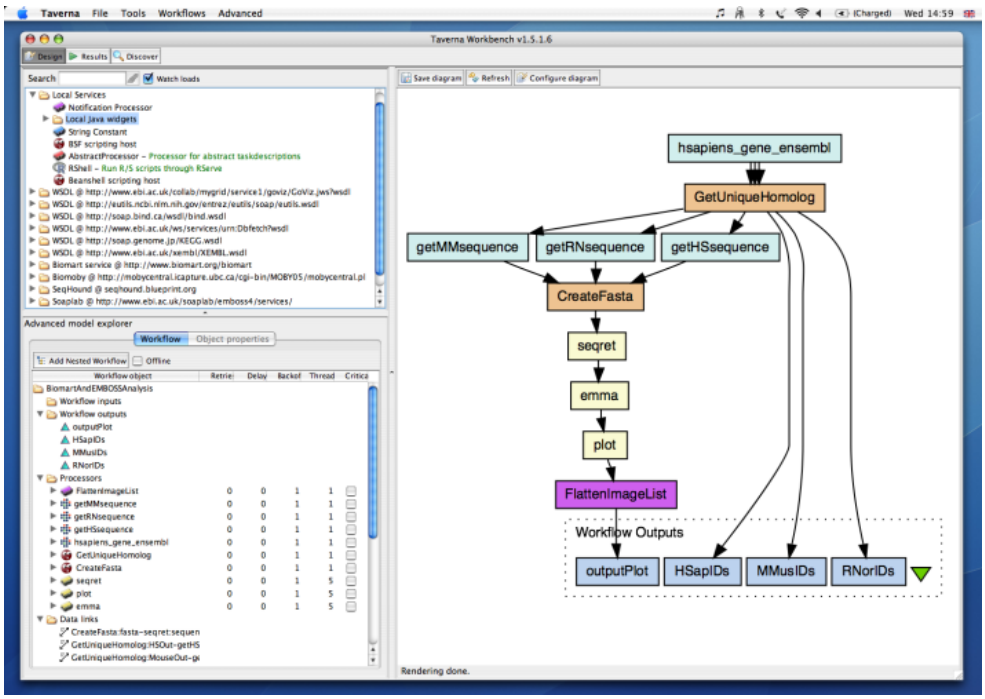


Figure 1.6: *Creación de servicios Grid con Taverna.*

Las herramientas de composición de servicios responden a una necesidad de flexibidad y modelización de la complejidad, sacrificando en muchos casos la facilidad de uso. De esta forma, el usuario con unos conocimientos avanzados de Informática y básicos de computación distribuida, puede modelar el comportamiento del sistema con un interfaz o lenguaje avanzado. El perfil de aplicaciones que suelen ser portado empleando estas herramientas es el de flujos de trabajos, que se discutirá más en profundidad en el Capítulo 3 de esta Tesis. Debajo de un compositor de servicios se tiene que encontrar una herramienta de alto nivel que gestione los trabajos en los que deriven las instrucciones introducidas. Por norma, es esta herramienta de alto nivel la que proporciona al usuario final un compositor de servicios, y a veces éste está integrado en un portal Grid, como es el caso de P-GRADE. Un ejemplo de lenguaje es Business Process Execution Language [FBS04] (BPEL), que proporciona métodos de definición y soporte tanto para flujos de trabajo como procesos de negocio. Otro ejemplo, esta vez de un interfaz que permite al usuario componer de formar gráfica servicios Grid complejos, es Taverna [HWS+06] (véase Figura 1.6).

Capítulo 2
Portado de Aplicaciones de Alta Productividad

> Ich habe gelernt, das Wort *unmöglich* mit gröβter Vorsicht zu sagen.
>
> Wernher von Braun

Las aplicaciones de alta productividad son aplicaciones de escasa complejidad en su defición, siendo el tipo de flujo de trabajos más sencillo, pero cuya ejecución eficiente y escalable supone un enorme desafío. En muchas ocasiones, no basta con disponer de un gran número de recursos para resolver las necesidades de estas aplicaciones. Se hace necesario una reducción de tiempos en alguna de las fases implicadas, a fin de que el tiempo de proceso total no exceda lo aceptable y pueda garantizarse la escalabilidad.

Este capítulo comenzará con una introducción a la Física de Fusión, para luego centrarse en el algoritmo escogido para su estudio y optimización (Sección 2.2). En la Sección 2.3 se hará un estudio del Estado del Arte de las aplicaciones de alta productividad. A continuación, en la Sección 2.4 se describirá el portado de la apliación y se justificará elección de las herramientas empleadas. Seguidamente, se analizarán los primeros resultados y se revisará el algoritmo proponiendo una mejora y validándola, en las Secciones 2.5 y 2.6 respectivamente. El capitulo finalizará con unas conclusiones en la Sección 2.7.

2.1. Física de Fusión

Aquella rama de la Ciencia que estudia el proceso mediante el cual, dos núcleos atómicos se unen para formar uno de mayor peso atómico (véase la Figura 2.1), se conoce como Física de Fusión. Este nuevo núcleo posee una masa inferior a la de los dos núcleos anteriores, lo que provoca que esa diferencia sea liberada en forma de energía. La cantidad de esta energía se corresponde con la famosa fórmula $E = mc^2$, en la que m es la diferencia de masa y c, la velocidad de la luz, fijada en 300.000

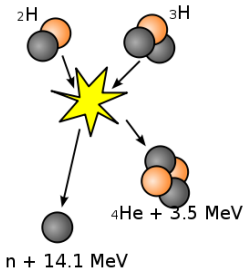


Figure 2.1: *Fusión deuterio-tritio.*

Km/s [Ein61]. Las estrellas son un ejemplo de fusión nuclear, pues es a ésta que deben su característico brillo.

Debido a su gran potencial, este proceso ha sido objeto de estudio por el Hombre desde hace más de 50 años no consiguiendo todavía una reacción controlada. La Investigación en este campo se ha centrado en el diseño de un reactor que permita soportar la enorme presión y temperatura que requiere una fusión nuclear. En la práctica, la energía con la que iniciar el proceso es demasiado alta como para considerar la fusión una fuente viable, amén de las dificultades para contener la reacción antes mencionadas. En esta Investigación se encuentran dos líneas principales según el tipo de confinamiento de la fusión:

- Inercial, que emplea partículas o rayos láser proyectados contra una unidad de combustible, provocando una ignición instantánea. Proyectos famosos que proceden de esta línea son el National Ignition Facility[1] (NIF) en Estados Unidos y el Laser MégaJoule[2] (LMJ) en Francia.

- Magnético, que contiene el combustible mediante un campo magnético. En su interior se aumenta la temperatura y presión formando lo que se conoce como plasma [Sta81].

La aplicación objeto del estudio de este Capítulo pertenece al ámbito de esta segunda línea, que se describirá en la siguiente Sección.

[1] https://lasers.llnl.gov/
[2] http://www-lmj.cea.fr/

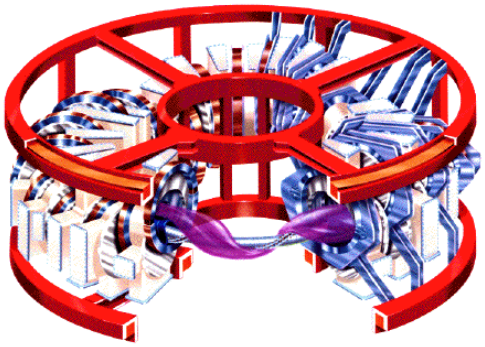


Figure 2.2: *Reactor Stellarator TJII del CIEMAT.*

2.2. De un Rayo a un Grid Computacional

Entre los reactores por confinamiento magnético se encuentran los Stellarators, de fabricación estadounidense. En particular, el Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas[3] (CIEMAT) posee un Heliac Flexible TJII, resultado de una colaboración con el Oak Ridge National Laboratory[4] (ORNL) y el Max-Planck-Institut für Plasmaphysik[5] (IPP). Uno de los problemas que se pretende abordar usando este reactor (véase la Figura 2.2) es la reducción de la energía necesaria para iniciar todo el proceso. Proceso que comienza una vez que el plasma se encuentra en el reactor y se disparara un haz de microondas que eleva su temperatura. Dicho haz tiene una frecuencia del rango del ciclotrón de iones o electrones o bien, cercana a uno de sus armónicos.

El calentamiento por resonancia electrón-ciclotrón (Electron Cyclotron Resonance Heating, ECRH) [LPSF77] se caracteriza por su escasa longitud de onda, permitiendo el estudio de sus propiedades empleando aproximaciones óptico-geométricas. Esto significa que un haz de microondas puede simularse mediante una gran cantidad de rayos. Si no hay una capa crítica en el plasma (del tipo corte o resonancia) cercano a la cintura del haz, es posible emplear una aproximación de campo lejano y realizar lo cálculos en un *cluster*. No obstante, si la cintura del haz está cercana a la capa crítica y el método de calentamiento emplea ondas de electrones del tipo Bernstein (Electron Bernstein Waves, EBW) [RBLD02], el número de rayos necesario será mayor. Al ser independientes todos los trazados de rayos, este problema

[3]http://www.ciemat.es/
[4]http://www.ornl.gov/
[5]http://www.ipp.mpg.de/

es idóneo para ser resuelto empleando el Grid [Cas05].

Cada uno de los rayos mencionados anteriormente puede ser trazado mediante la aplicación propuesta para ser portada al Grid. Esta aplicación, llamada *Truba*, que en ruso se escribe *Tpy6a* y define a un tipo de trompeta ucraniana, es fruto de una colaboración entre el CIEMAT y el General Physics Institute[6] (GPI). *Truba* puede ir encapsulada en un trabajo Grid, procesando así un rayo distinto del haz de microondas lanzado al interior del reactor de fusión [CTea04]. La aplicación, programada en Fortran, tarda en procesar un rayo 9 minutos en un Pentium 4 a 3,20 GHz. Además, su tamaño es 1,8 MB, el de los archivos de entrada (información del rayo y geometría del TJ-II) es 70 KB y el de los de salida, 549 KB. El sistema final, al tratarse de un trazado masivo de rayos, tendría por nombre *MAssive RAy TRAcing in Fusion Plasmas* (MaRaTra, Trazado Masivo de Rayos en Plasmas de Fusión). Si bien el *cluster* del CIEMAT había llegado a procesar del orden de 50 a 100 rayos en un tiempo razonable para poder realizar el mayor número de experimentos posible, las necesidades científicas hablaban de una magnitud muy superior (1000 rayos como mínimo) de tal forma que suponga una mejora en el proceso de Investigación. Esta mejora en el proceso de Investigación se traduce entonces, en trazar el mayor número de rayos el mayor número de veces posible, a fin de encontrar rápidamente el lugar idóneo desde donde disparar el haz de microondas y calentar el plasma del reactor consumiendo la menor energía. El Grid debería recoger el testigo de la Computación Cluster y aumentar así el número de rayos procesados, reduciendo a la par el tiempo total, tal y como detalla en el artículo del Apéndice A.5.

2.3. Aplicaciones de Alta Productividad

Las aplicaciones de alta productividad, encuentran en la Computación Grid una plataforma apropiada para su ejecución. Son muchas las áreas científicas, tales como la Bioinformática, la Dinámica de Fluidos Computacional o la Física de Partículas, en las que se verifica este hecho. Son de especial interés para la presente Tesis las aplicaciones de barrido de parámeros, que conllevan la ejecución de un gran número de tareas, realizando cada una un cálculo diferenciado sólo por los parámetros que ésta recibe.

Así como la estructura de este tipo de aplicaciones denota simplicidad, es la propia naturaleza del Grid la que aparece como contrapunto. Uno de los problemas más complejos con el que la comunidad Grid debe lidiar, teniendo como fin la ejecución eficiente el tipo de aplicaciones descritas anteriormente, es el hecho de que los Grids presentan condiciones cambiantes e impredecibles. Estas condicio-

[6] http://www.gpi.ru/

nes son la alta tasa de error y la disponibilidad dinámica de recursos, así como la carga y el coste. Para salvar estas dificultades se recurre a la planificación adaptativa [AAF+01, BWC+03, VD03] como solución clara al dinamismo antes expuesto. Además, es posible incrementar el rendimiento y la tolerancia de fallos de a través de la migración de trabajos a recursos más apropiados [LARS01]. Esta migración se basaría en eventos generados dinámicamente tanto por el Grid como por la aplicación ejecutada.

2.4. Trazado Masivo de Rayos en Plasmas de Fusión

Partiendo de la base que el portado de *Truba* iba a emplear la infraestructura perteneciente al Proyecto EGEE, se solicitó al CIEMAT una versión de la aplicación, para la arquitectura Intel. Posteriormente, se evaluaron las tecnologías disponibles y compatibles con la infraestructura existente: El Workload Management System (WMS) de EGEE y Globus GridWay. Esta evaluación, detallada en los artículos aportados en los Apéndices A.2 y A.4, no sólo fue teórica, sino que también se realizaron dos implantaciones de la arquitectura del sistema final.

2.4.1. ¿EGEE WMS o Globus GridWay?

El WMS de EGEE es el resultado de proyectos anteriores al propio EGEE (Datagrid[7], CrossGrid[8]). Su versión actual recibe el nombre de gLite [BCD+08] y está basada en el *middleware* LCG-2 [DMD+04], del cual hereda sus elementos constituyentes y funcionalidades. La versión LCG-2 era la única disponible cuando el portado de la aplicación tuvo lugar y por tanto, fue objeto del estudio previo. La arquitectura de WMS de EGEE, representada en la Figura 2.3, es altamente centralizada y cada funcionalidad es suministrada por casi una máquina difrente, en el momento que está concevido como un servicio de red. Sus componentes son: el User Interface (UI), que es desde donde el usuario envía los trabajos; el Resource Broker (RB), que está basado en Condor-G y aprovecha casi todas sus funcionalidades; el Computing Element (CE) y los Worker Nodes (WN), que son el frontend y los nodos de un *cluster* respectivamente, tal y como está establecido en la rígida configuración de la arquitectura de LCG-2; el Storage Element (SE), empleado para almacenar los ficheros del trabajo; el Logging and Bookkeeping service (LB), que registra eventos relativos al trabajo [CLS04].

Después de que el usuario comience a utilizar el Grid autenticándose desde el UI y que los parámetros del trabajo sean definidos, el RB recibe los ficheros de en-

[7] http://eu-datagrid.web.cern.ch/
[8] http://www.crossgrid.org/

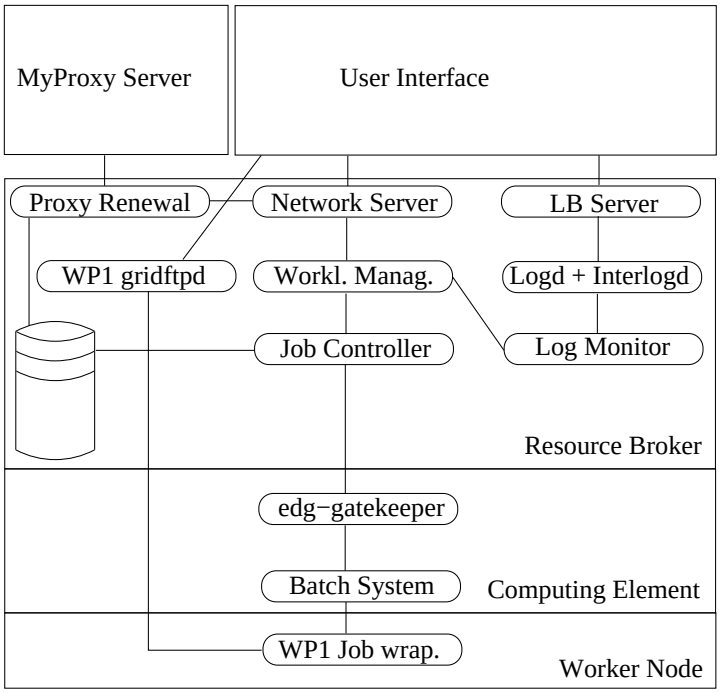


Figure 2.3: *Arquitectura del WMS de EGEE.*

trada a través de un *input sandbox*. El RB contacta con el Servicio de Información, donde se publican los recursos disponibles (CE y SE). Si fuera necesario, el RB también consultará el Servicio de Catálogo. Entonces, el Job Submission Service recibe una versión expandida de la descripción el trabajo, resultado de las decisiones de planificación tomadas por el RB (proceso de *matchmaking*), y manda el trabajo al CE elegido. Mientras tanto, el RB envía a ese CE el *input sandbox* y la información de contacto. El trabajo pasa entonces a un WN donde es ejecutado. Cuando éste finaliza, el *output sandbox* es enviado de vuelta al RB y de ahí, al usuario.

GridWay [HML04] se encuentra por encima de los servicios Globus, gestionando la ejecución de trabajos y asignación de recursos, permitiendo además que dicha ejecución sea desatendida, fiable y eficiente en Grids *débilmente acoplados* [LMH05] formados por recursos Globus. Se consideró este metaplanificador como opción porque la infraestructura de LCG-2 emplea Globus (con unas ligeras modificaciones), posibilitando por tanto su uso, tal y como se detalla en los artículos aportados en los Apéndices A.1 y A.3. Su arquitectura altamente modular, representada en la Fi-

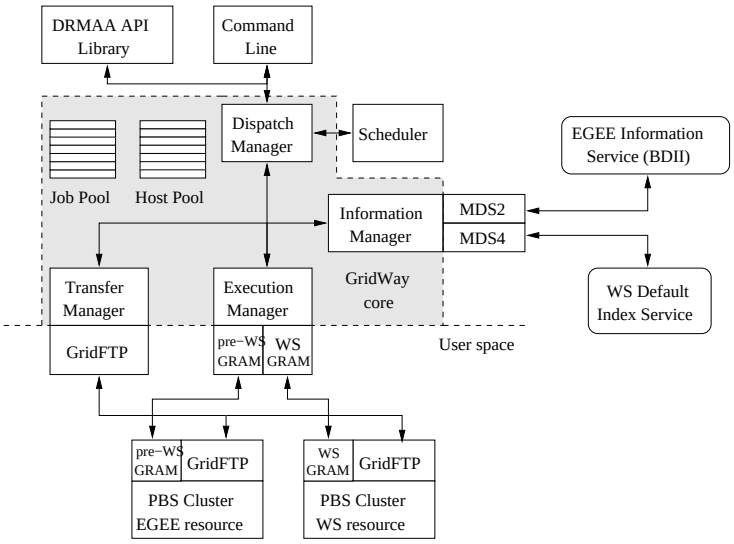


Figure 2.4: *Arquitectura de GridWay.*

gura 2.4, está formada por el GridWay Daemon (GWD) y diferentes Middleware Access Drivers (MADs) para acceder a diferentes servicios Grid (información de recursos, ejecución de trabajos y transferencia de archivos), todos ellos en el mismo equipo pues GridWay está concebido como una herramienta cliente. GridWay planifica trabajos de forma dinámica, y además los migra bajo petición o de forma oportunista.

Después que el GWD recibe del usuario la descripción el trabajo, se procede a seleccionar el recurso. Con la decisión ya tomada, se cargan los MADs necesarios para la ejecución y la transferencia. La primera fase se llama *prolog* y en ella se prepara el sistema remoto, creando el directorio de trabajo y enviando los archivos de entrada. Posteriormente viene el *wrapper* que es cuando tiene lugar la verdadera ejecución del trabajo. Finalmente, durante el *epilog*, los ficheros de salida son traídos de vuelta y el directorio remoto es borrado. Como aspecto adicional, coviene decir que GridWay favorece el despliegue de metaplanificadores a nivel de organización que proporcionan soporte para usuarios repartidos en diferentes instancias de planificación. De esta forma, existe una instancia de planificación por cada organización y así todas las instancias compiten entre ellas por los recursos disponibles.

Se procedió a comparar el WMS de EGEE con GridWay y el punto de partida fueron las capacidades de planificación. Ambos procesan los trabajos en modo FIFO

y planifican de forma dinámica, ofreciendo una forma de filtrar y evaluar recursos basada en atributos dinámicos, a través de expresiones de *ranking*. Los nombres de estos atributos en LCG-2 son los mismos que los obtenidos del servicio de información. En GridWay se ha añadido una capa de abstracción para que los atributos sean genéricos, ofreciendo así otra forma más de desacoplamiento. Mientras el RB sólo accede a servidores con Berkeley Database Information Index[9] (BDII) y sólo procesa el esquema Grid Laboratory Uniform Environment (GLUE) [And04], los diferentes MADs de GridWay permiten el acceso a los sistemas de información más comunes. Considerando tanto la ejecución como la transferencia, y la adopción de los interfaces y protocolos basados en Web Services (WS), los dos operan con los servicios pre-WS de Globus, pero sólo GridWay puede acceder a los servicios WS [HML06b]. GridWay proporciona mecanismos de migración oportunista, esto es, en cada ciclo de planificación evalúa los beneficios de migrar trabajos a nuevos recursos disponibles (añadidos o liberados) mediante la comparación de expresiones de *ranking*. LCG-2 no ofrece esta funcionalidad y el *ranking* sólo afecta al envío de trabajos. Evaluando la detección de degradaciones en el rendimiento, GridWay toma en cuenta el tiempo de suspensión en los sistemas remotos y solicita una migración cuando se supera un umbral predefinido. Además, los trabajos son enviados incluyendo un sistema ligero de monitorización. El trabajo migrará cuando el monitor detecte que se no recibe el porcentaje de CPU mínimo esperado. Ninguno de estos mecanismos de detección de degradaciones en el rendimiento está implantados en LCG-2. La monitorización en la arquitectura de LCG-2 es suministrada por el LB, que sólo registra estados básicos de los trabajos y los mezcla con los originados por otros componentes. Con GridWay, una aplicación puede tomar decisiones sobre la selección de recursos a medida que su ejecución evoluciona, modificando sus expresiones de requisitos y *ranking*, así como solicitando una migración. En LCG-2 por el contrario, estas expresiones sólo son fijadas al principio. En LCG-2, los mecanismos de contabilidad distribuida son proporcionados por PBS Event Logging (APEL) [ABCea04]. GridWay ofrece funcionalidades de contabilidad local a través de Berkeley Database [OBS99].

El siguiente aspecto que se consideró en el análisis fue la detección y recuperación de errores. Éste es de vital importancia para el portado de esta aplicación, pues uno de los requisitos consiste en que ninguno de los rayos podía perderse (su ejecución debe ser siempre exitosa). LCG-2 incorpora los mecanismos de detección de errores de Condor-G [MFF+05]. Por el otro lado, GridWay detecta la cancelación de un trabajo (cuando el código de salida del trabajo no ha sido especifado), caída el sistema remoto y desconexión de la red (ambos cuando el *polling* del trabajo falla). En todos estos casos, GridWay solicita una migración para el trabajo [HML06a]. El

[9] https://twiki.cern.ch/twiki/bin/view/EGEE/BDII/

WMS de LCG-2 ofrece mecanismos de *checkpointing* incluyendo un API que permite adaptar las aplicaciones para guardar el estado del proceso (representado como una lista de variables y valores) en cualquier momento durante la ejecución del trabajo, y así reiniciar el proceso desde esos datos cuando fuera necesario. Si un trabajo falla, el WMS automáticamente lo replanifica a fin de enviarlo a otro recurso compatible, para reiniciar su ejecución desde el último estado guardado. El usuario puede además recuperar esos datos para realizar un reenvío manual posterior y decidir desde que estado debería retomarse la ejecución. En GridWay, los ficheros de *checkpoint* o reinicio corren a cargo del usuario. La migración consiste en reiniciar la ejecución del trabajo en un nuevo recurso. Si no se suministran los ficheros de *checkpoint*, el trabajo se ejecutará desde el principio. Estos *checkpoints* son recogidos periódicamente, bien por la máquina cliente, bien por un servidor aposta. Pero también puede fallar el sistema que ejecuta el planificador local. GridWay almacena de forma persistente su estado con el objetivo de recuperar o reiniciar los trabajos una vez que el sistema rearranca. LCG-2 delega en Condor-G, que almacena todo aquel estado relevante para cada trabajo ennviado en la cola del planificador [FTL+02].

También la funcionalidad a nivel de interfaz de usuario fue analizada. Tanto GridWay como el RB de LCG-2 pueden gestionar trabajos sencillos y con dependencias. Para estos últimos, LCG-2 se apoya en Condor *DAGMan* [ABCea04] y con GridWay, el usuario puede sincronizar trabajos. Además, LCG-2 permite trabajos interactivos. Por otro lado, GridWay permite manejar trabajos paramétricos (mismo ejecutable pero distinta entrada). La sincronización de trabajos en LCG-2 corre a cargo del usuario, teniendo que implantar un mecanismo de espera activa, si bien la dependencia de trabajos, tal y como se comentó antes, se realiza a través de *DAGMan*. Así, por cada Grafo Acíclico Dirigido (Direct Acyclic Graph, DAG) se crea un proceso *DAGMan* que será enviado a Condor-G. Ambas tecnologías otorgan al usuario un control total de sus trabajos, con el anãdido en GridWay de que éstos se pueden migrar y sincronizar. Además, GridWay ofrece diversas implantaciones del estándar DRMAA (véase Sección 1.5) mientras que EDG WMS API[10], suministrada por LCG-2, no es directamente un estándar.

El último aspecto que fue considerado es la instalación y la configuración. Ambas tecnologías son modulares. El RB de LCG-2 es un servicio de red que se apoya sobre un gran número de dependencias externas (requiere otros servicios además de Globus): Condor-G, MySQL, etc. GridWay se apoya en el *middleware* de Globus en el lado del cliente, pero además puede ser usado como un elemento de construcción para arquitecturas más complejas que implementan servicios más avanzados. EGEE anima a instalar LCG-2 sólo en máquinas con Scientific Linux[11], aunque se

[10] http://www.to.infn.it/grid/workload_management/apiDoc/edg-wms-api-index.html
[11] https://www.scientificlinux.org/

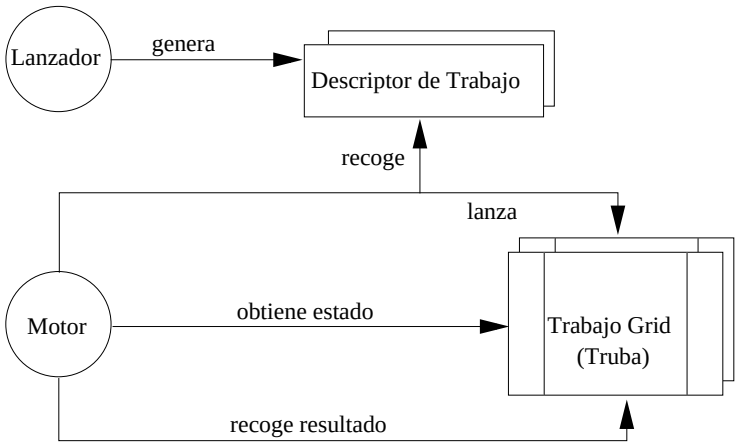


Figure 2.5: *Funcionamiento de MaRaTra.*

están usando otras distribuciones. Analizando los mecanismos de seguridad, LCG-2 emplea los del GSI de Globus Toolkit[12] y además implementa VOMS, donde un usuario tiene un nombre y un rol dentro de una organización virtual. VOMS permite un control de grano fino del uso de los recursos tanto para la organización de los usuarios como la propietaria de los recursos [ACea05].

2.4.2. Arquitectura del Sistema

Antes de continuar con la evaluación de ambas herramientas de alto nivel, se procedió a diseñar la arquitectura del sistema, representada en la Figura 2.5. En ella se distinguen dos entidades principales: el lanzador y el motor. El lanzador se encarga de generar descriptores de trabajo en función de los rayos a analizar. En estos descritores de trabajo se mantiene el ejecutable (*Truba*) y en los ficheros de entrada, el de la geometría del reactor. El resto de parámetros cambia totalmente. Los descriptores de trabajo pueden reducirse al de un trabajo paramétrico, en cuyo caso, el lanzador sólo deberá informar al motor acerca del número de trabajos totales. El motor, por su lado, se encarga de recoger los descriptores de trabajo y elevarlos al metaplanificador para su envío al Grid. Finalmente, obtendrá los resultados de las ejecuciones pero para ello, el motor deberá conocer el estado de los trabajos.

Para la implantación con el WMS de EGEE, se empleó el lcg2.1.69 User Interface C++ API. Si bien su especificación dispone que se pueden definir colecciones de

[12]http://www.globus.org/toolkit/security/

trabajos, un *bug* hacía fallar todo el sistema cuando se sobrepasara un número indefinido de trabajos, por lo que fue necesario reimplantar esa funcionalidad a partir de trabajos sencillos. Además, se precisó un mecanismo de espera activa para conocer el estado de cada trabajo. A la finalización de los mismos, se debía hacer una llamada explícita a una función de recogida de ficheros de salida.

Con GridWay se empleó DRMAA C y el proceso fue más sencillo. La creación de trabajos paramétricos se pudo realizar empleando las funciones del estándar sin ningún problema. La obtención de los resultados se hizo más simple pues la implantación de DRMAA de GridWay ofrece la funcionalidad de sincronización de trabajos.

2.5. Primeras Experiencias

Durante los primeros experimentos, publicados en los artículos de los Apéndices A.1 y A.3, se empleó la infraestructura perteneciente a la organización virtual de prueba de la Federación Suroeste de EGEE (SWETEST VO). En la Tabla 2.1 puede encontrarse más información sobre los centros y máquinas que participaron. Todos los centros españoles están conectados por la Red Española Académica y de Investigación[13] (RedIRIS), cuyos enlaces están representados en la Figura 2.6. Los trabajos fueron enviados desde la Universidad Complutense de Madrid y, en el caso de LCG-2, el RB estaba ubicado en el IFIC.

Table 2.1: *Recursos Grid de SWETEST VO empleados.*

Centro	Procesador	Velocidad (GHz)	Nodos	Gestor Local
CESGA	Intel Pentium III	1.1	46	PBS
IFAE	Intel Pentium 4	2.8	11	PBS
IFIC	AMD Athlon	1.2	127	PBS
INTA-CAB	Intel Pentium 4	2.8	4	PBS
LIP	Intel Xeon	2.8	25	PBS
PIC	Intel Pentium 4	2.8	172	PBS
USC	Intel Pentium III	1.1	100	PBS

La Tabla 2.2 muestra un resumen del rendimiento obtenido con los dos sistemas de metaplanificación. T_{exe} y T_{xfr} son los tiempos medios de ejecución y el de transferencia respectivamente, representados en la comparativa por centros de la Figu-

[13]http://www.rediris.es/

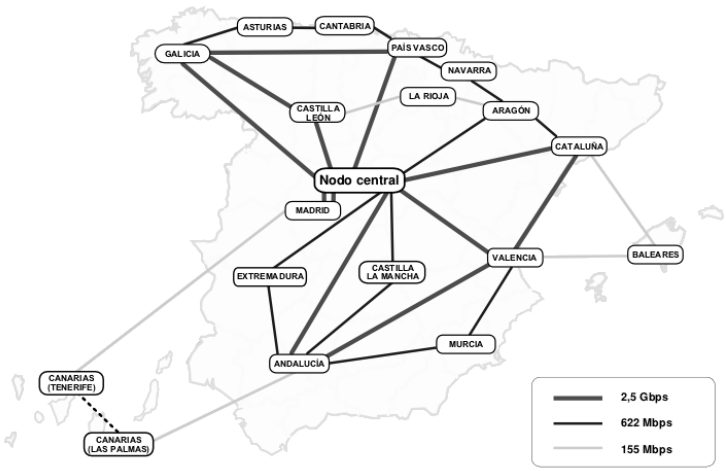


Figure 2.6: *Infraestructura de RedIRIS.*

ra 2.7 y la Figura 2.8. Como puede observarse, GridWay presenta un mayor tiempo de transferencia, debido a los trabajos adicionales correspondientes a las fases de *prolog* y *epilog* [HML04]. No obstante, estos experimentos fueron realizados con una versión de GridWay previa a la 4.7, en la cual se redujo este tiempo añadido. Por otro lado, la desviación estándar de métricas del rendimiento puede ser un buen indicador de la heterogeneidad tanto de los recursos Grid como de sus interconexiones, tal y como se explica en los artículos aportados en los Apéndices A.1 y A.3. Finalmente, el *overhead* (retardo introducido por la propia tecnología) de GridWay muestra los beneficios de la ligereza en su concepto y sus mecanismos de detección de la degradación en el rendimiento.

Table 2.2: *Métricas de rendimiento con ambas tecnologías.*

	T_{exe} (m)		T_{xfr} (m)		T	Prod.	Ovh.
	Media	Desv.	Media	Desv.			
LCG-2	30.33	11.38	0.42	0.06	195	15.38	1.82
GridWay	36.80	16.23	0.87	0.51	120	25.00	0.52

El WMS de EGEE tardó 195 minutos (3,25 horas) en ejecutar 50 trabajos, alcanzando una productiviad de 15,38 trabajos/hora. GridWay tardó 120 minutos (2 ho-

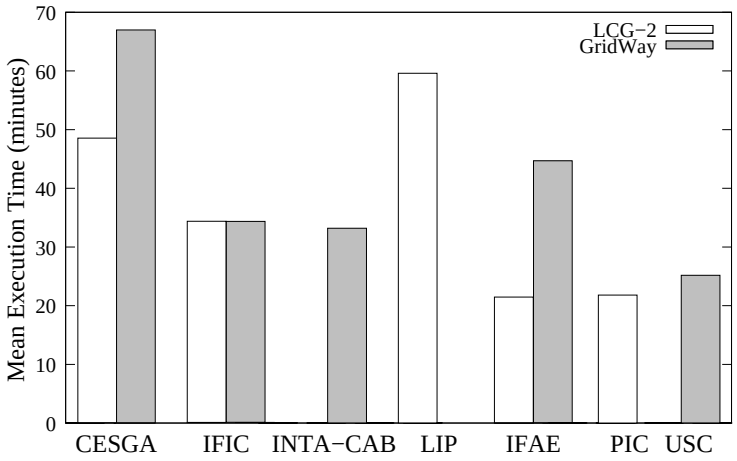


Figure 2.7: *T_{exe} medio por centro.*

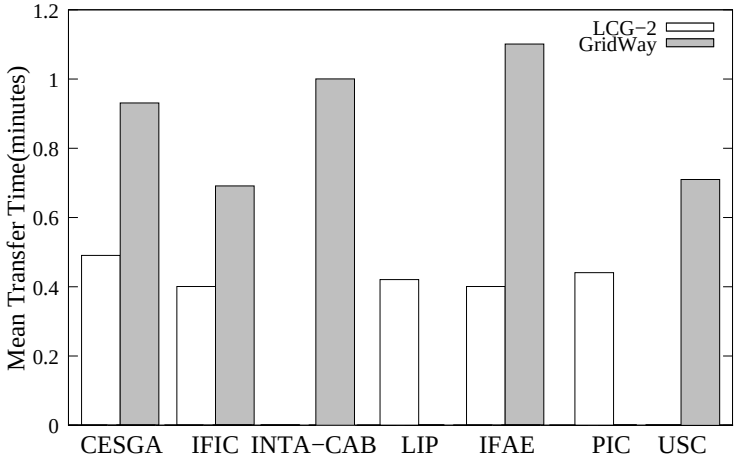


Figure 2.8: *T_{xfr} medio por centro.*

ras) en ejecutar la misma carga, con una productividad de 25 trabajos/hora. la conclusión es que GridWay aprovecha más los recursos disponibles debido a sus capacidades superiores de planificación en recursos dinámicos. De hecho, durante los experimentos con el WMS de EGEE, se evidenciaron bastantes problemas descritos en el análisis teórico. El RB de LCG-2 no ofrece migración oportunista ni detección

de la degradación, por lo que los trabajos son asignados a recursos ocupados.

Adicionalmente, se calculó el nivel de paralelismo [HML05] obtenido por ambas soluciones. Esta métrica se puede calcular empleando la siguiente expresión:

$$U = \frac{T_{exe}}{T}, \tag{2.1}$$

siendo T_{exe} la suma de los tiempos de ejecución de los trabajos y T, el tiempo total. Así, deducimos que el nivel de paralelismo obtenido por GridWay (14,91) fue más alto que el del WMS de EGEE (6,89).

De todas formas, no todos los trabajos terminaron exitosamente durante su primer envío. Con el WMS de EGEE, 31 trabajos necesitaron ser reenviados. Con GridWay sólo falló 1 trabajo, pero 21 migraciones tuvieron lugar, producidas principalmente por unos tiempos de suspensión (espera en cola) excesivos y el descubrimiento de mejores recursos cuando el trabajo estaba asignado a uno excesivamente inferior.

En [MHL06] fue propuesta una metodología para analizar el rendimiento de Grids computacionales en la ejecución de aplicaciones de alta productividad. Este modelo facilita la comparación de diferentes plataformas en los siguientes términos: rendimiento asintótico (r_∞), que es el mayor número de tareas ejecutadas por segundo, y la longitud de rendimiento medio ($n_{1/2}$), que es el número de tareas necesarias para obtener la mitad del rendimiento asintótico. Una caracterización de primer orden empleando estos parámetros puede expresarse como:

$$n(t) = r_\infty t - n_{1/2}, \tag{2.2}$$

Entonces, el rendimiento del sistema, trabajos completados por segundo, puede definirse como un número finito de tareas:

$$r(n) = n(t)/t = \frac{r_\infty}{1 + n_{1/2}/n}, \tag{2.3}$$

donde n es el número de trabajos. Los parámetros del modelo, r_∞ y $n_{1/2}$, son obtenidas con un ajuste lineal de los resultados experimentales de la ejecución de las aplicaciones.

La Figura 2.9 y la Figura 2.10 muestran el rendimiento experimental obtenido con las dos tecnologías, así como el predicho por la Ecuación (2) y Ecuación (3). Con el WMS de EGEE, r_∞ fue 0,0051 trabajos/segundo (18,19 trabajos/hora) y $n_{1/2}$ fue 8,33. Con GridWay r_∞ fue 0,0079 trabajos/segundo (28,26 trabajos/hora) y $n_{1/2}$ fue 1,92. Se puede deducir de los diferentes valores de $n_{1/2}$ que GridWay necesita menos trabajos para obtener la mitad del rendimiento asintótico, debido a una asignación más temprana de trabajos a recursos.

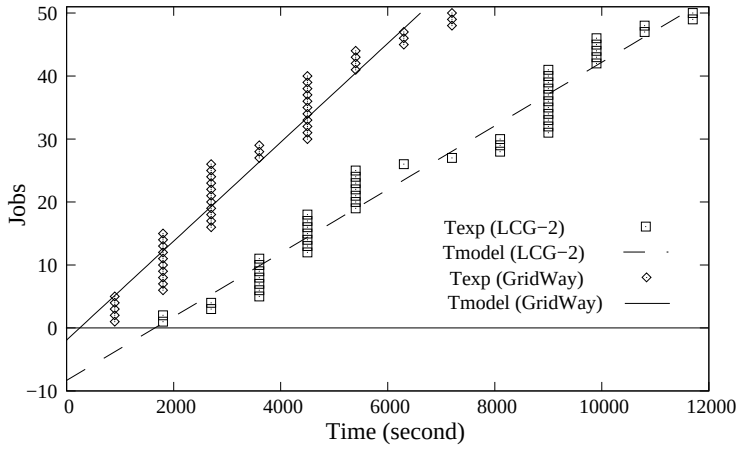


Figure 2.9: *Valores de r_∞ y $n_{1/2}$ para las dos plataformas.*

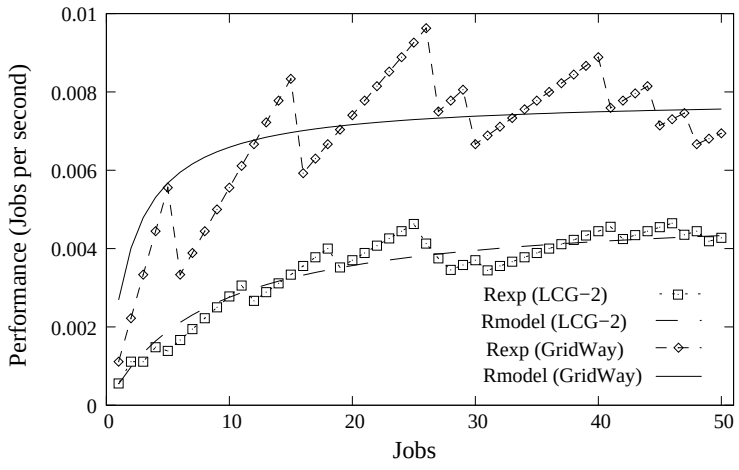


Figure 2.10: *Rendimiento exponencial y predicho.*

Con estas primeras experiencias, se ha demostrado que GridWay es una buena alternativa al RB de LCG-2, tanto para usuarios, desarrolladores de aplicaciones, como administradores Grid. GridWay ofrece compatibilidad a las aplicaciones que empleen sistemas gestores que a su vez, implanten el estándar DRMAA. El interfaz de comandos es similar al de los gestores de recursos locales para enviar, terminar,

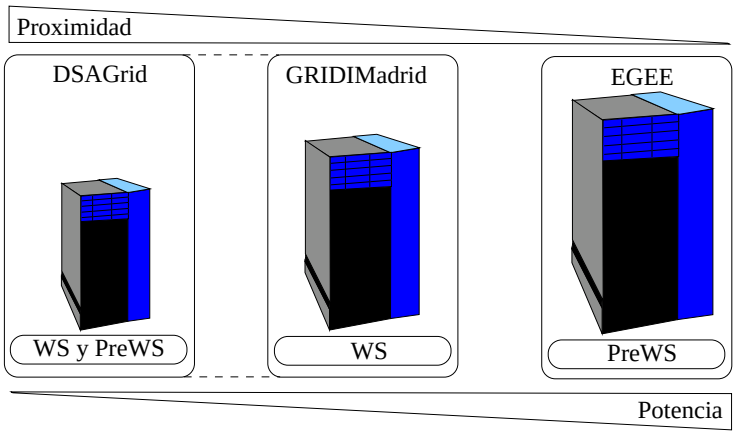


Figure 2.11: *DSAGrid, GRIDIMadrid y EGEE.*

migrar, monitorizar y sincronizar trabajos (sencillos, paramétricos y con dependencias). Adicionalmente, el despliegue y mantenimiento de GridWay no es solamente rápido y sencillo, sino que además, se puede realizar sobre un amplio abanico de plataformas. GridWay introduce menos *overhead* y ofrece más productividad que el WMS de EGEE. Esto es básicamente debido a la reducción de fases en el envío de trabajos, así como sus mecanismos de migración oportunista y detección de la degradación en el rendimiento, que mejoran considerablemente el uso de los recursos. No obstante, en la comparación realizada no se trató la gestión de datos, por no ser necesaria para la aplicación.

2.6. Optimización de la Ejecución

Los primeros resultados de la implantación de MaRaTra fueron exitosos, consolidando a GridWay como la herramienta de alto nivel a usar para su portado. No obstante, el prototipo necesitó ser revisado, puesto que se debía incrementar el número de rayos en proceso.

La primera mejora planteada consistió en extender la infraestructura Grid empleada. Si bien EGEE posee numerosos recursos, en más de una ocasión puede encontrarse un alto grado de ocupación de los mismos ya que existen otras aplicaciones y experimentos en producción. Por otro lado, las políticas existentes en los recursos disponibles sobre el número máximo de trabajos simultáneos ejecutándose, suelen limitar bastante la infraestructura operativa. Es por esto que se aprovechó la ca-

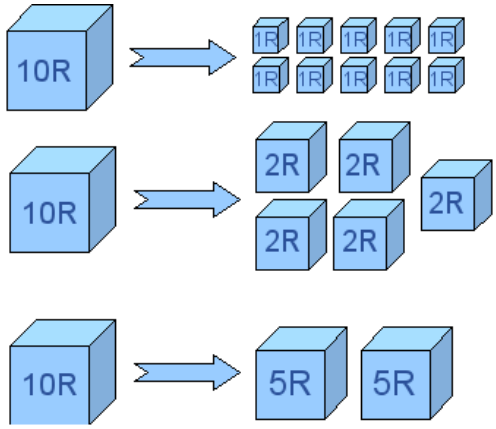


Figure 2.12: *Uso de chunks en MaRaTra.*

pacidad de GridWay para interactuar con diferentes *middleware* Grid básicos (véase la Sección 2.4.1) y a los recursos accedidos por la aplicación Grid se sumaron DSA-Grid y GRIDIMadrid, descritos en las Secciones 1.2.1 y 1.2.2, respectivamente. Estas infraestructuras Grid no disponen de tanta potencia de cálculo como la de EGEE, pero sí permiten liberarla en parte de su carga de trabajos. Esto es posible principalmente gracias a la proximidad de estas nuevas infraestructuras Grid, en las que el trabajo no terminará antes necesariamente, pero sí las transferencias de archivos, tanto de entrada como de salida, tal y como se representa en la Figura 2.11.

La siguiente mejora fue resultado del estudio de los tiempos de espera de los trabajos en el recurso remoto y fue presentada en [VPHM+07] a fin de mostrar a la Comunidad de Fusión sus beneficios. De las primeras experiencias, se observó que dicho tiempo permanecía prácticamente invariable. Como este tiempo se aplica a cada trabajo enviado, se consideró el agrupamiento de ejecuciones de *Truba* en el mismo trabajo o *chunk*. Esta mejora, representada en la Figura 2.12, redunda entonces en una reducción tanto del número de trabajos a gestionar como del tiempo total.

Se probó con diferentes tamaños de *chunk* para procesar 101 y 1001 rayos en varios experimentos. La elección de estos tamaños, si bien tentativa, responde a las limitaciones impuestas al tiempo de ejecución de cada trabajo por los recursos de computación pertenecientes a la infraestructura. En cada *chunk* se ejecuta un trabajo que procesa un número especifico de rayos. El número total de *chunks* según las consideraciones tomadas está especificado en la Tabla 2.3 y los tiempos totales obtenidos, representados en la Figura 2.13. Para 101 rayos, se puede apreciar que

Table 2.3: *Número de chunks enviados según su tamaño.*

Rayos/*chunk*	101 Rayos	1001 Rayos
5	21	201
10	11	101
15	7	67

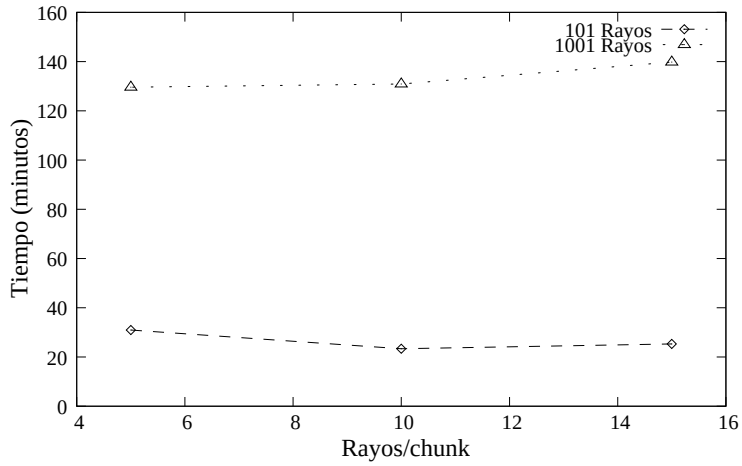


Figure 2.13: *Tiempos totales con diferentes chunks.*

el número ideal de trabajos agrupados es 10. Para 1001 rayos, así como existe una tendencia alcista, la elección de 10 rayos difiere escasamente en tiempo total de la de 5 rayos, no siendo así para el número de trabajos totales, los cuales son reducidos considerablemente. La reducción del número de trabajos simultáneo simplifica la gestión de los trabajos, con un inferior número de transferencias simultáneas que no colapsaría el ancho de banda. No obstante, ligada a esta pérdida de *overhead* se encuentra también la pérdida del nivel paralelismo, al reducir el número de tareas simultaneas.

2.7. Conclusiones

Queda constatado que la metaplanificación es un paso necesario en la evolución del *middleware* Grid. En este capítulo, además de introducir el ámbito de la

aplicación portada al Grid, la Física de Fusión, se ha realizado una comparativa entre dos implantaciones que se podían ajustar a las necesidades de la misma: El WMS de EGEE y GridWay. Este análisis no sólo ha sido teórico, sino que además, ambas plataformas han sido puestas a prueba en un entorno de producción como es la infraestructura Grid de EGEE.

Se ha demostrado que GridWay ofrece una alternativa de metaplanificación al RB de LCG-2, distribuida por EGEE, para usuarios, desarrolladores de aplicaciones y administradores Grid. GridWay ofrece compatibilidad a aplicaciones con sistemas de gestión de recursos distribuidos que implantan el estándar DRMAA. El interfaz de comandos es similar a los encontrados en los sistemas de gestión recursos locales para enviar trabajos, migrarlos, monitorizarlos, sincronizar (tanto trabajos sencillos como paramétricos y con dependencias). Además, el despliegue y el mantenimiento no sólo es sencillo y rápido, sino que además, es posible emplear un amplio abanico de plataformas. Finalmente, GridWay produce un *overhead* inferior, así como ofrece una mayor productividad. No obstante, LCG-2 tiene también otros componentes que no fueron considerados, como es el caso de la gestión de datos.

Una vez que se optó por GridWay para portar definitivamente la aplicación y seguirla en su ciclo de desarrollo, se procedió a optimizar el sistema final. Esta optimización consistió, por un lado, en aprovechar la funcionalidad de GridWay que permite interoperar con otras infraestructuras. Por el otro lado, se adoptó el uso de *chunks* para reducir el número total de trabajos y con ello, la suma de los tiempos de espera en recurso remoto. Finalmente, la viabilidad de esta solución para 101 y 1001 rayos, con un tamaño tentativo de trabajos, fue comprobada de forma exitosa realizando varios experimentos.

La herramienta, disponible de forma abierta para la comunidad de Física de Fusión, ha tenido una gran aceptación. Una prueba de ello es que se está ejecutando en producción. Además, se están recibiendo peticiones de mejora y creando aplicaciones que emplean MaRaTra internamente.

Capítulo 3

Portado de Flujos de Trabajos

> Quanti conigli può allevare una coppia in un anno?
>
> Leonardo di Pisa (Fibonacci)

A medida que la Computación Grid se está consolidando y extendiendo su aplicación a muchas áreas científicas, la complejidad de las aplicaciones involucradas ha ido incrementándose. El concepto de flujo de trabajos aparece para satisfacer estas necesidades mediante la automatización de los procedimientos de transferencia de datos y ejecución de trabajos. Si en el capítulo anterior se presentaba un algoritmo que respondía al más sencillo de los flujos de trabajos, en éste la complejidad aumentará no sólo en el algoritmo estudiado, sino también en las optimizaciones requeridas para proporcionar eficiencia y superar las limitaciones impuestas por el hardware.

Este capítulo comenzará con una introducción a la Bioinformática, campo científico al que pertenece el algoritmo cuya aplicación será portada al Grid. Ambos serán explicados seguidamente en la Sección 3.2, para luego desarrollar el Estado del Arte de los flujos de trabajos en la Sección 3.3. El proceso de portado de la aplicación será explicado en la Sección 3.4 y sus resultados experimentales, en la Sección 3.5. Nuevamente y como una de las aportaciones de esta Tesis, el algoritmo necesitó una revisión para optimizar su funcionamiento. Así, en la Sección 3.6 se estudiarán las posibles optimizaciones existentes en la Literatura, para luego discutir en la Sección 3.7, los resultados derivados de la aplicación de la replicación y la aglomeración. Al detectar una serie de paramétros de entrada que alteraban el tiempo total de proceso, se confeccionó un modelo de ejecución que se mostrará en la Sección 3.8 y que permite una elección óptima de dichos parámetros de optimización. Finalmente, el capítulo termina con una Sección de conclusiones (Sección 3.9).

3.1. Bioinformática

Como área científica, la Bioinformática es de reciente creación (2002). Ésta puede definirse como la aplicación de métodos científicos en computación, con el objetivo de analizar datos experimentales de origen biológico, así como la simulación de sistemas biológicos. Los datos analizados suelen provenir de moléculas relevantes para la vida, generalmente de origen genómico y proteómico. Profundizando más en las aplicaciones de la Bioinformática, se puede encontrar también la minería de datos (*data mining*), así como el desarrollo de herramientas informáticas que faciliten y agilicen los procesos antes indicados.

Dada la heterogeneidad de las disciplinas que intervienen en la Bioinformática, son apreciables varias líneas de Investigación principales:

- **Anotación de genomas.** Se trata de un proceso de marcado de genes, así como de otras características detectables en secuencias de ADN. El primer sistema se remonta a 1995 [WF95].

- **Análisis de la expresión génica.** Existen diversas técnicas físicas para realizar este análisis, entre ellas: *microarrays* de ADN, secuenciación de EST, y análisis en serie de la expresión génica. Todas ellas están sujetas a sesgos y ruidos ocasioandos por la propia medición. Es por ello que la principal línea de trabajo consiste en desarrollar herramientas estadísticas para aislar los datos relevantes en estudios de alto volumen de procesamiento [MHV+02].

- **Análisis de secuencias.** Cantidades ingentes de secuencias de ADN pertenecientes a diversos organismos han sido decodificadas y guardadas en bases de datos desde 1977. Su análisis permite dilucidar los genes que codifican ciertas proteínas, así como las propias secuencias reguladoras. De esta forma, la comparación de genes entre especies (o dentro de la misma especie) permite descubrir similitudes entre funciones de proteínas y relaciones entre las propias especies [CG84].

- **Genómica comparativa.** El objeto de estudio es en este caso la correspondencia entre genes o bien, entre otras características genómicas de una población de diferentes organismos. De esta forma, se pueden construir los denominados mapas intergenómicos que permiten rastrear los procesos evolutivos que separan a dos genomas [KPD+04].

- **Medición de la biodiversidad.** La biodiversidad es el complemento genómico total de todas las especies que conviven en un medio ambiente concreto.

La labor computacional que entraña su análisis está repartida entre el almacenamiento de las características en bases de datos, el análisis de esa información, su compartición, y finalmente una simulación de la dinámica poblacional [HBWI07].

- **Biología evolutiva computacional.** Consiste en el estudio del origen ancestral de las especies y su evolución a lo largo del tiempo. Se suelen emplear algoritmos genéticos y su gran meta es construir el cada vez más complejo árbol filogenético de la vida [FM67].

- **Análisis de la regulación.** Se define como la orquestación de eventos que parten de una señal extracelular (por ejemplo, una hormona) y estudian su repercusión en la actividad de una proteína o un conjunto de ellas [Elg96].

- **Acoplamiento proteína-proteína.** A medida que se descubren nuevas estructuras tridimensionales de proteínas, la predicción de posibles interacciones ha ido cobrando interés. La solución se presenta en forma de simulaciones, basándose exclusivamente en la forma de estas estructuras, no necesitando entonces experimentos reales de interacción proteína con proteína [CGVC04].

- **Análisis de imagen de alto rendimiento.** El campo de la Biomedicina se distingue por producir grandes cantidades de imágenes con un alto contenido de información. La Bioinformática suministra por consiguiente, tecnologías de computación para acelerar o automatizar el proceso, así como analizar las imágenes suministradas [SNW+07].

- **Análisis de la expresión de proteínas.** En este caso, la Bioinformática se plantea como una ayuda tanto a la tecnología de *microarrays* de proteínas como a la espectrometría de masas de alto rendimiento [Liu07].

- **Predicción de la estructura de las proteínas.** En esta línea, el objetivo es predecir la secuencia de aminoácidos de una proteína. Esto se logra a través de otra secuencia, que es la de nucleótidos sobre el gen que la codifica. Hasta ahora, la Bioinformática ha provisto heurísticas que se tornan útiles en la mayor parte de los casos [Yua05].

- **Modelado de sistemas biológicos.** Se centra en simulaciones de subsistemas celulares, por lo que la computación juega una gran baza. La meta consiste en analizar las conexiones, muy complejas, entre los procesos celulares. Conceptos extraídos de esta línea de Investigación son la vida artificial y la evolución virtual [BL].

- **Análisis de mutaciones en el cáncer.** Así como la reordenación genómica en células afectadas por el cáncer es compleja e impredecible, si es posible realizar análisis individuales de las secuecias. La Bioinformática entonces produce sistemas que gestionan de forma automática la gran cantidad de datos extraídos de las secuencias analizadas, así como desarolla algortimos para comparar los resultados [HSV+].

3.2. Un Grid Computacional para comparar Proteínas

Tal y como se explica en el artículo del Apéndice A.7, la aplicación objetivo fue propuesta para su portado al Grid por el Centro Nacional de Investigaciones Oncológicas[1] (CNIO). Su nombre es *cd-hit* [LG06] (*Cluster Database at High Identity with Tolerance*) y está compuesta por un conjunto de herramientas que realiza *clustering* de proteínas. Este proceso consiste en eliminar las secuencias redundantes de una base de datos de proteínas para que sólo las representativas permanezcan. El *clustering* de proteínas tienen muchos usos, tales como la clasificación de familias de proteínas, análisis de dominios, organización de grandes bases de datos de proteínas o mejorar el rendimiento en sus búsquedas. Pero de entre los usos de *cd-hit*, el más famoso es la generación de conjuntos de datos UniRef para UniProt[2], que es el mayor catálogo de información estructurada de proteínas del mundo.

El algoritmo implantado por *cd-hit* está representado en la Figura 3.1. Inicialmente, una herramienta llamada *cd-hit-div* realiza divisiones en la base de datos de proteínas. A la primera división, considerada como la representativa, se le aplica la herramienta *cd-hit* con el fin de compararse consigo misma (la tarea A en la Figura 3.1). La salida es entonces comparada con el resto de divisiones mediante la herramienta *cd-hit-2d* (las tareas B). La primera división resultado de la última operación es considerada a continuación como la representativa (la tarea A' en la Figura 3.1) y el proceso comienza de nuevo hasta que no queden más divisiones. Una vez que la última comparación ha concluido, todas las salidas de las diferentes llamadas a *cd-hit* son consolidadas empleando la herramienta *clstr_merge.pl*.

La razón por la cual se solicitó el portado de *cd-hit* empleando la Computación Grid radica en el tamaño de las bases de datos procesadas. A medida que el número de proteínas en éstas se incrementa, más inviable se vuelve todo el proceso en una sola máquina, debido a los requisitos de memoria y al tiempo total de ejecución. Por otro lado, la complejidad asociada al algoritmo requiere un gran número de recursos de computación, tanto para la ejecución de sus trabajos como de las transferencias

[1] http://www.cnio.es/
[2] http://www.pir.uniprot.org/database/DBDescription.shtml

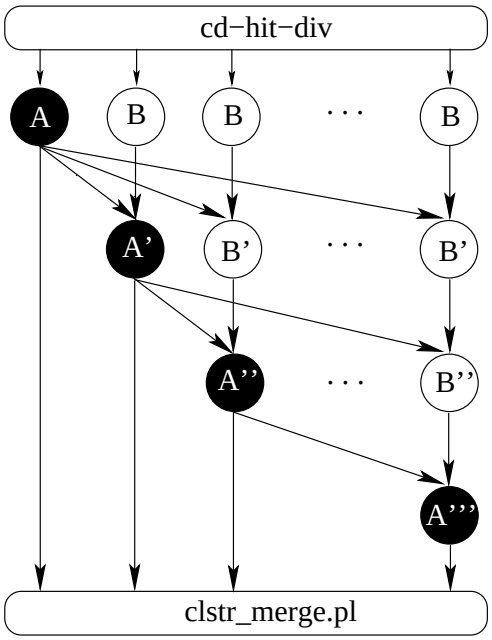


Figure 3.1: *Algoritmo implantado por cd-hit.*

de sus archivos, justifica el uso de una gran infraestructura Grid.

3.3. Flujos de Trabajos

Los flujos de trabajos (*workflows*) caracterizan un trabajo computacional como un conjunto de tareas con unas determinadas relaciones de dependencia entre sí. Estas relaciones de dependencia pueden estar condicionadas por el orden de ejecución, los datos utilizados, o bien otros factores. Podemos distinguir muchos tipos de flujo de trabajos, en base al aspecto que estemos considerando.

El flujo de trabajos más sencillo que puede encontrase es el **Barrido de Parámetros** [Pin02], tipo al que pertenece la aplicación del Capítulo 2. De hecho, las aplicaciones de **Barrido de Parámetros** están englobadas en las de Alta Productividad. Como se recordará, todas las tareas de este tipo de flujos se ejecutan en paralelo sin existir una relación de precedencia entre las mismas.

Adicionalmente, nos encontramos también con las **Cadenas Adaptativas** [PCM+07] en las que sus tareas tienen una sola entrada y una sola salida, si bien dichas tareas

pueden ser un conjunto en otras.

Dentro de los flujos de forma arbórea, con tareas que se van ramificando, distinguimos:

- **Árboles de Entrada** [Pin02, MB05] y **Árboles de Entrada Densos** [MB05], en los que el número de tareas en paralelo se va reduciendo a medida que navegamos por el flujo de trabajos.
- **Árboles de Salida** [Pin02, MB05] y **Árboles de Salida Densos** [MB05], en los que por el contrario, el número de tareas en paralelo va aumentando.

Si consideramos cómo se definen las **Condiciones de Precedencia** entre tareas, podemos distinguir entre **Acotados** [Pin02] y **Arbitrarios** [Pin02]. Ahondando en las condiciones de precedencia y en particular, el sentido en el que éstas confieren una forma al flujo de trabajos, distinguimos entre:

- **Grafos Dirigidos Acíclicos** [YB05], en los que todas las tareas son ejecutadas como máximo una vez.
- **Grafos Dirigidos No Acíclicos** [YB05], en los que por el contrario, se pueden dar ciclos de tareas.

3.4. Arquitectura del Sistema

Con el objetivo de portar la aplicación al Grid, se ha distinguido en dos tipos de tareas. En el primer tipo, el trabajo ejecuta *cd-hit-2d* y seguidamente, *cd-hit* sobre la división de la base de datos tomada. En el segundo tipo, el trabajo sólo ejecuta *cd-hit-2d*. Tanto la división de la base de datos como la consolidación final son realizadas localmente. De la Figura 3.1 se extrae que las tareas de un cierto nivel no pueden comenzar hasta que el primer trabajo del nivel anterior haya conluido, con lo que las dependencias globales de cada tarea son fácilmente distinguibles, así como el perfil de flujo de trabajos del propio sistema.

Se empleó el metaplanificador GridWay que ya había sido usado en el campo de la Bioinformática [HBML04]. En particular, se usó su implantación del DRMAA[3]. Así como en el anterior capítulo se describieron las capacidades de gestión de carga de GridWay, a continuación y a modo de justificación de su uso, se describirá GridWay empleando la taxonomía propuesta por [YB05]. Un estudio más extendido con una pequeña comparativa puede encontrarse en el artículos de los Apéndices A.6 y A.7.

[3] http://drmaa.org/

GridWay, cuya arquitectura ya ha sido expuesta en la Sección 2.4.1, es capaz de gestionar flujos de trabajos basados en grafos dirigidos acíclicos, donde cada nodo es una tarea cuyo inicio está sujeto a la finalización de otras tareas. Además, GridWay permite estructuras avanzadas de flujo como bucles y *branches* gracias a su implantación del DRMAA. Considerando la especificación de flujos de trabajos, GridWay emplea un *modelo abstracto* puesto que el flujo es definido sin referencias a recursos Grid específicos. Al concretizar flujos abstractos, GridWay sigue un *esquema dinámico* puesto que utiliza información estática y dinámica de los recursos. La planificación se realiza en tiempo de ejecución, siempre teniendo en cuenta los requisitos de cada tarea y las expresiones de *ranking*. Dentro de este esquema, GridWay sigue la llamada *planificación just in-time* en la que las decisiones sólo se toman durante la ejecución de la tarea. Además, también se considera la información sobre la ejecución de las tareas. La arquitectura de GridWay, tal y como se define en [YB05], es *centralizada* porque las decisiones son tomadas por un único planificador central.

Por el otro lado, la toma de decisiones es *local* porque la información de cada tarea sólo es considerada cuando éstas se planifican, ya que GridWay resuelve sus dependencias y comienza a trabajar a nivel de tarea. Una vez que una tarea (o nodo del grafo) es asignada a un recurso, el ejecutable y los ficheros de entrada son transferidos a la máquina remota y la ejecución tiene lugar. Cuando la tarea finaliza, los ficheros de salida son traídos de vuelta, y GridWay verifica si este evento libera de dependencias a otras tareas de forma que el proceso comience de nuevo [HML04]. GridWay ofrece mecanismos de transferencia automática. En particular, se toma una aproximación centralizada puesto que los datos intermedios son tranferidos entre recursos a través de un servidor de *checkpoint*, que sirve para reiniciar el flujo de trabajos en caso de fallo.

La tolerancia a fallos en GridWay está concebida a nivel de tarea, tomando en cuenta posibles fallos del tipo cortes de red o caídas tanto de la máquina remota como de la local. Siguiendo la nomenclatura de [YB05], GridWay aplica las siguientes técnicas: *reintento* (intenta la ejecución o la transferencia de archivos en el mismo recurso en caso de fallo), *recurso alternativo* (envía la tarea fallida a un recurso alternativo) y *checkpoint/reinicio* (las tareas fallidas son movidas transparentemente a otros recursos).

3.5. Primeras Experiencias

Como primera base de datos a procesar se empleó una parte de la RefSeq[4], distribuída para el National Center for Biotechnology Information (NCBI) de Es-

[4]http://www.ncbi.nlm.nih.gov/RefSeq/

tados Unidos. Su tamaño era de 435 MB y almacenaba 504.876 proteínas. El número de tareas resultantes y por ende, el tamaño de los ficheros de entrada, están sujetos a las divisiones que se hayan realizado inicialmente a la base de datos, tal y como se muestra en la Tabla 3.1. Por su parte, tanto *cd-hit* como *cd-hit-2d* ocupan 1,1 MB.

Table 3.1: *Tamaño de archivos y número de tareas por cada división.*

Divisiones	Tamaño Medio	Tareas
10	44 MB	45
12	36,5 MB	66
14	31,5 MB	91
16	27,5 MB	120
18	24,5 MB	153
20	22 MB	190
22	20 MB	231

Los experimentos, cuyos resultados fueron publicados en los artículos aportados en los Apéndices A.6 y A.7, fueron realizados en recursos del proyecto EGEE dedicados a las actividades de Biomedicina, debido a que, como ya se ha precisado en el anterior capítulo, GridWay puede ser usado en esta infraestructura [VPHML06]. Los recursos conformantes de la infraestructura de los experimentos están detallados en la Tabla 3.2. Los trabajos fueron enviados desde la Unversidad Complutense de Madrid en diferentes momentos del día, durante diferentes días de la semana, y a lo largo de julio del 2007.

Considerando la ejecución de cada nodo del flujo de trabajos de *cd-hit*, observamos que el comportamiento del algoritmo depende del número inicial de las divisiones realizadas a la base de datos. Así como su incremento favorece el nivel de paralelismo, la razón ejecución/transferencia de archivos empeora. Este hecho queda reflejado en la Figura 3.2, donde se representan los tiempos medios de CPU (T_{cpu}), transferencia (T_{xfr}) y espera en cola (T_{que}) para cada división realizada sobre la base de datos. En cualquier caso, el tiempo que una tarea espera en la cola del sistema remoto no está relacionado con el número de divisiones, puesto que depende directamente de la carga que tenga el recurso. Esta carga suele ser alta debido a que la infraestructura EGEE se encuentra en producción. Así, el tiempo de espera en cola ocupa un porcentaje muy alto del tiempo total, si no consideramos los tiempos de transferencia, de cada tarea en todos los experimentos.

Con el objetivo de analizar el impacto de las consideraciones antes realizadas para el tiempo total, se definió el *tiempo total esperado* (T_{exp}) sin contar con los tiempos

Table 3.2: *Recursos de EGEE durante los primeros experimentos.*

Centro	País	Procesador	Nodos	Velocidad
BIFI	ES	Intel P.IV	56	3.2GHz
CESGA	ES	Intel P.III	16	500MHz
CGG	FR	Intel P.III	58	1.2GHz
CIEMAT	ES	Intel Xeon	226	3.2GHz
GRIF	FR	Intel P.IV	14	2.8GHz
JINR	RU	Intel P.D	30	2.8GHz
L.-HEP	UK	Intel P.IV	374	3GHz
PNPI	RU	Intel P.IV	60	3GHz
RAL	UK	Intel P.IV	62	2.8GHz
RALPP	UK	Intel P.III	1064	1GHz
ScotGRID	UK	Intel Xeon	6	2.8GHz
SINP	RU	Intel Xeon	94	2.8GHz

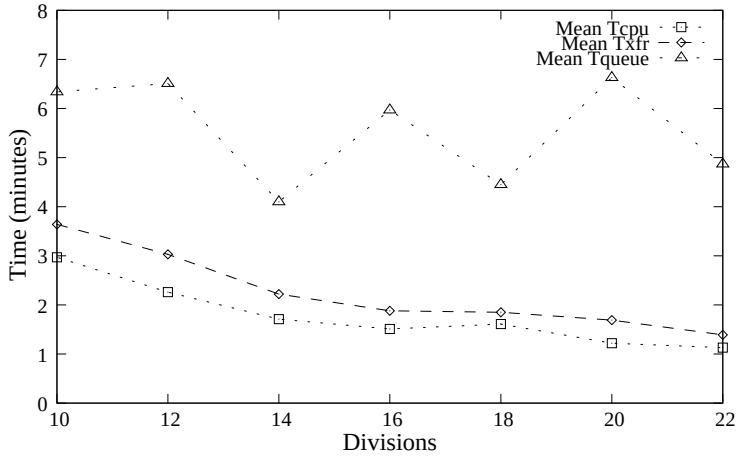


Figure 3.2: *Tiempos medios de CPU (T_{cpu}), transferencia (T_{xfr}) y espera en cola (T_{queue}) del flujo de trabajos para diferentes divisiones de la base de datos.*

de espera en cola y los fallos. Partiendo del hecho que la finalización de un nivel del flujo de trabajos está sujeta a la ejecución del nodo más significantivo:

$$T_{exp} = N \cdot (T^A_{cpu} + T^A_{xfr}), \tag{3.1}$$

donde N es el número de divisiones de la base de datos, y T^A_{cpu} y T^A_{xfr} son los

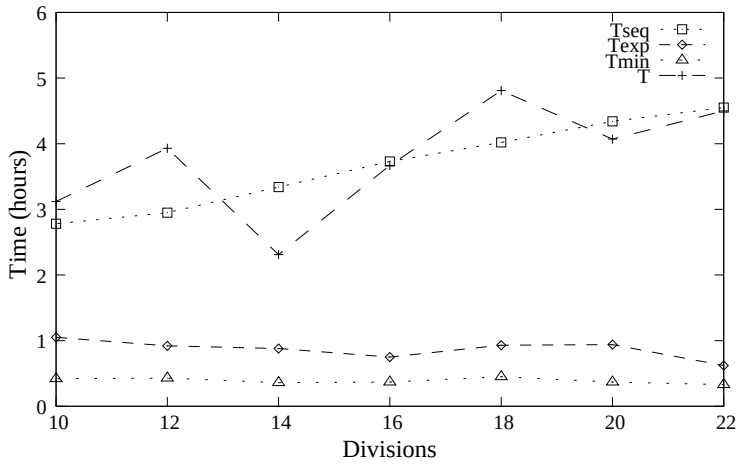


Figure 3.3: *Tiempos de ejecución del flujo de trabajos según las divisiones realizadas.*

tiempos medios de CPU y transferencia de las tareas sombreadas en la Figura 3.1. Además una estimación a la baja del tiempo total sería: $T_{min} = N \cdot T^A_{cpu}$. Finalmente, es posible realizar una comparación con la ejecución secuencial del flujo de trabajos (T_{seq}) tal que:

$$T_{seq} = N \cdot T^A_{cpu} + T^B_{cpu} \cdot \sum_{n=2}^{N}(n-1), \tag{3.2}$$

donde T^B_{cpu} es el tiempo de CPU de las tareas blancas en la Figura 3.1. Los tiempos de CPU de la Ecuación 3.2 fueron medidos en la máquina más rápida de la infraestructura Grid (Intel Pentium IV 3,2 GHz).

La Figura 3.3 muestra los tiempos anteriormente citados, así como el tiempo experimental (T) obtenido de la ejecución del flujo de trabajos con difrentes divisiones de la base de datos. Como puede observarse, el tiempo de ejecución Grid es similar al obtenido en una única máquina, y diferente del predecido por la Ecuación 3.1. Esto es debido principalmente al *overhead* impuesto por los sistemas de gestión de recursos remotos, tal y como se muestra en la Figura 3.2.

El tiempo total del flujo de trabajos depende también del número de veces que un trabajo es replanificado empleando otro recurso. En este caso, estas replanificaciones fueron debidas a errores de ejecución (por ejemplo, fallos del *middleware*) o tiempos de suspensión excesivos. El número de trabajos replanificados en cada experimento se muestra en la Figura 3.4. Consecuentemente, la influencia de los dos factores antes citados puede observarse claramente en los valores para 14 divisiones en la

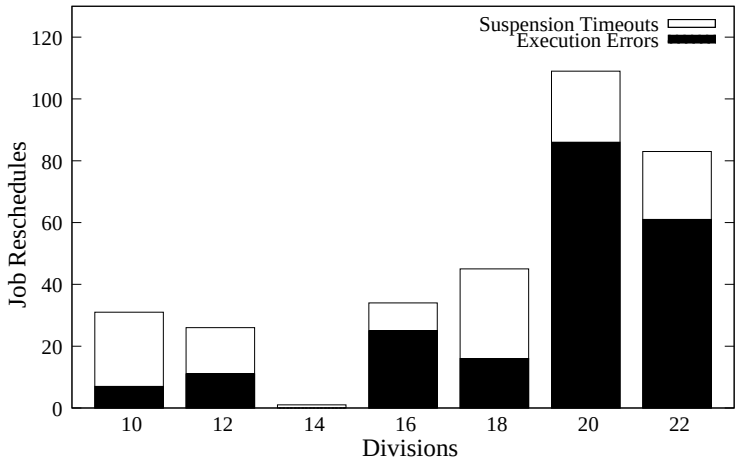


Figure 3.4: *Número de trabajos replanificados en cada experimento.*

Figura 3.5.

Es interesante analizar el *speed-up* potencial que se podría obtener con este tipo de aplicaciones, en las que el nivel de paralelismo no se mantiene constante (el número de tareas disminuye en cada nivel del flujo de trabajos). De esta forma se consideró: el *speed-up* (S) del flujo de trabajos, el obtenido sin los tiempos de espera en cola ni fallos de trabajos (S_{exp}), y un extremo superior (S_{max}) calculado con T_{min}. Estos valores están representados en la Figura 3.5. Como se discutió anteriormente el *speed-up* obtenido es muy limitado. Finalmente, los tiempos de transferencia de archivos también imponen una reducción significante del *speed-up* esperado, si se compara éste con S_{max}.

3.6. Alternativas a la Optimización de la Ejecución

Así como la Computación Grid sirvió para procesar separadamente las divisiones hechas sobre la base de datos, la eficiencia esperada se vio limitada dramáticamente. Esto se debió a la propia naturaleza del Grid: dinamismo (tiempo de espera en cola), heterogeneidad y una tasa de fallos alta. Con el objeto de optimizar la ejecución del flujo de trabajos, se estudiaron las diferentes mejoras existentes en la Literatura.

Entre los métodos de ejecución eficiente, se distinguió entre heurísticas clásicas, heurísticas contemporáneas y adicionalmente, heurísticas basadas en la tolerancia a

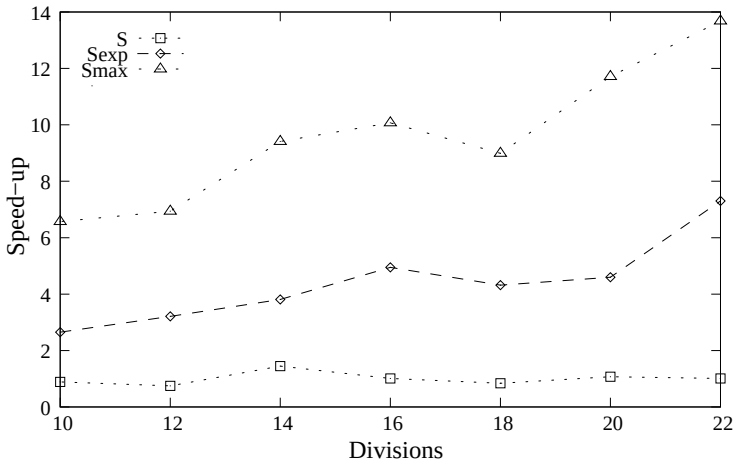


Figure 3.5: *Speed-up del flujo de trabajos según las divisiones realizadas.*

fallos. Todas éstas se detallan a continuación.

3.6.1. Heurísticas Clásicas

Las siguientes heurísticas han sido extraídas de [Pin02]. Por un lado, nos encontramos con heurísticas deterministas, tales como:

- **Mayor Tiempo de Proceso primero (*Longest Processing Time first*, LPT)**: Asigna al principio los m trabajos más largos a las m máquinas disponibles. Después de esto, cuando una máquina es liberada, el trabajo más largo de aquellos no procesados, es asignado a dicha máquina. Esta heurística intenta colocar los trabajos más cortos al final de la planificación, donde pueden ser usados para equilibrar las cargas.

- **Camino Crítico (*Critical Path*, CP)**: Asigna la mayor prioridad al trabajo que se encuentra al comienzo de la lista de trabajos más largad e un grafo de precedencias. Aquí se pueden distinguir dos tipos de trabajos: los críticos, que pertenecen al mencionado camino crítico y los holgados, que son el resto.

- **Mayor Número de Sucesores primero (*Largest Number of Successors first*, LNS)**: El trabajo con mayor número de sucesores en el grafo de precedencia tiene la mayor prioridad. En el caso de los flujos de trabajo del tipo *intree*, las heurísticas CP y LNS son equivalentes.

- **Último Trabajo Flexible primero (*Least Flexible Job first*, LFJ)**: Cada vez que una máquina se libera, un trabajo disponible que puede ser procesado por el menor número de máquinas es asignado a la misma. Esta heurística puede ser combinada con una en la que se considera la flexibilidad desde el punto de vista de la máquina.

Adicionalmente, existen heurísticas no deterministas, a saber:

- **Mayor Tiempo de Proceso Estimado (*Longest Expected Processing Time*, LEPT)**: Los trabajos con mayor tiempo de proceso estimado son los primeros en ser elegidos para su asiganción a máquinas disponibles.
- **Menor Tiempo de Proceso Estimado (*Shortest Expected Processing Time*, SEPT)**: Los trabajos con menor tiempo de proceso estimado son los primeros en ser elegidos.
- **Menor Varianza primero (*Lowest Variance first*, LV)**: Los trabajos con menor varianza en su tiempo de ejecución (obtenido en un escenario determinista) son los primeros en ser elegidos.

3.6.2. Heurísticas Contemporáneas

Existen tres tipos de heurísticas que entran en esta clasificación: Planificación en Lista, Aglomeración y Replicación.

Planificación en Lista

Dentro de esta heurística, las prioridades son asignadas a los trabajos bien de forma estática, bien de forma dinámica [SS94]. Variantes dentro de esta categoría son:

- **Tiempo Heterogéneo de Finalización Temprano Primero (*Heterogeneous Earliest Finish Time*, HEFT)**: Las tareas se planifican minimizando su tiempo de finalización empleando un método de inserción [THW02].
- **Camino Crítico en un Procesador (*Critical Path On a Processor*, CPOP)**: Destaca una máquina para la ejecución exclusiva de tareas pertenecientes al camino crítico [THW02].
- **Planificación y Asignación en Burbuja (*Bubble Scheduling and Allocation*, BSA)**: Primero se serializa un grafo de tareas, después se asignan todas las tareas a un procesador [KA00].

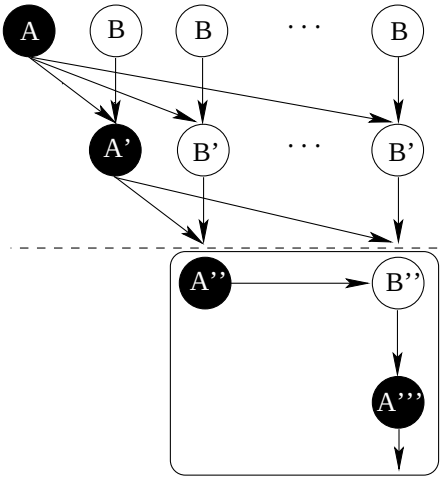


Figure 3.6: *Heurística de aglomeración: un conjunto de tareas es ejecutado localmente con el objetivo de contrarrestar los retardos ocasionados por las comunicaciones.*

- **Planificación de Nivel Dinámico (*Dynamic Level Scheduling*, DLS)**: Difiere la planificación cuando la tarea seleccionada está lista [SL93].
- **Árboles de Precedencia de Nodos Críticos (*Critical Nodes Parent Trees*, CNPT)**: Considera el tiempo de ejecución más temprano de la tarea [HJ03].
- **Asignación Heterogénea en Nivel Equivalente (*Iso-Level Heterogeneous Allocation*, ILHA)**: Asigna a cada procesador un número de tareas proporcional a su potencia computacional [OVY02].

Aglomeración

Al aplicar esta heurística, los trabajos son agrupados con el fin de reducir considerablemente los retardos debidos a las comunicaciones [BA04]. La Figura 3.6 ofrece una representación visual de esta heurística. El estudio de esta heurística está publicado en el artículo del Apéndice A.6.

Replicación

Algunos trabajos pueden ser replicados para ser ejecutados simultáneamente en diferentes recursos, con el fin de incrementar la posibilidad de mejora en el *speed-up*. Esta heurística está descrita gráficamente en la Figura 3.7 y su estudio, publicado en los artículos de los Apéndices A.6, A.8 y A.9.

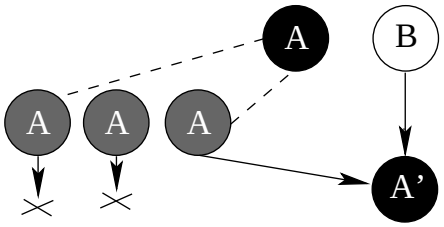


Figure 3.7: *Heurística de replicación: un conjunto de tareas es replicado para ser ejecutado en diferentes recursos, con el fin de incrementar la posibilidad de mejora en el speed-up.*

- **Duplicación Inversa de Tareas Críticas (*Critical Nodes Parent Trees*, CNPT)**: Considera el tiempo de ejecución más temprano de la tarea [HJ04].
- **Heurística de Duplicación Bottom-up y Top-down (*Bottom-up Top-down Duplication Heuristic*, BTDH)**: Asume que el tiempo de comienzo de una tarea puede reducirse eventualmente mediante la replicación de todas las tareas precedentes necesarias [CR92].
- **Tareas Críticas Precedentes Heterogéneas con Duplicación Rápida (*Heterogeneous Critical Parents with Fast Duplicator*, HCPFD)**: Realiza la replicación considerando el tiempo ocioso dejado por la tarea seleccionada en una máquina dada [HJ05].
- **Algortimo de Duplicación basado en el Camino Crítico (*Critical Path based Full Duplication Algorithm*, CPFD)**: Replica todos las posibles antecesores de la tarea considerada [AK98].

3.6.3. Heurísticas basadas en la Tolerancia a Fallos

Este tipo de heurísticas, basadas en [YB05] y compatibles entre sí, tienen como objetivo ofrecer capacidad de tolerancia a fallos al sistema. Por un lado, distinguimos aquellas aplicadas a nivel de tarea:

- **Reintento**: La ejecución se vuelve a probar en el mismo recurso donde se originó el error.
- **Recurso Alternativo**: La tarea fallida es enviada a otro recurso para su ejecución.
- ***Checkpoint***: La tarea fallida es movida de forma transparente a otros recursos.

- **Replicación**: Explicada en la Sección 3.6.2.

Por otro, existen huerísticas a nivel del propio flujo de trabajos:

- **Tarea Alternativa**: La tarea fallida es ejecutada de una forma diferente.
- **Redundancia**: Equivalente a la descrita en la Sección 3.6.2.
- **Manejo de Excepciones por parte del Usuario**: El usuario especifica el procedimiento a seguir en caso del fallo de una tarea.
- **Flujo de Trabajos de Rescate**: Un nuevo flujo de trabajos conteniendo las tareas fallidas es generado, a partir de información estadística.

3.7. Implantación de la Optimización

Tras estudiar las heurísticas antes descritas, se optó por implantar inicialmente la replicación. Dada la forma del flujo de trabajos, era la que prometía una mayor optimización. Adicionalmente, se escogió la aglomeración, no como una optimización aplicable de forma individual, sino como un complemento de la anterior heurística.

3.7.1. Heurística de Replicación

La heurística de replicación, representada en la Figura 3.7, se aplicó tal y como se describe en los artículos de los Apéndices A.6, A.8 y A.9. Con el objeto de reducir el efecto de los fallos y las esperas en cola, se crearon tareas suplementarias para ciertos nodos del flujo de trabajos, persiguiendo aumentar de esta forma las posibilidades de una rápida ejecución de dichos nodos.

Ahondando en la heurística escogida, se puede observar que existen diferentes variantes de esta estrategia, dependiendo de los nodos en los que se aplica. La más sencilla consiste en replicar todas las tareas. También existe otra variante que implica replicar aquellas tareas que conforman el camino crítico. Una última variante más compleja es aquella en la que la replicación afecta a aquellas tareas que superan un umbral de bloqueo prefijado. Se define el valor de bloqueo como el número de nodos del flujo de trabajos que dependen de la ejecución del nodo considerado, tal que:

$$b_{i,j} = \begin{cases} ST(N-i) & \text{if} \quad i = j \\ (i-1) + ST(N-i) & \text{if} \quad i \neq j \end{cases} \tag{3.3}$$

siendo i la columna y j el nivel correspondiente al nodo en el flujo de trabajos. N es el número de niveles y $ST(n)$ es el árbol de bloqueos (*blocking subtree*, ver Figura 3.8)

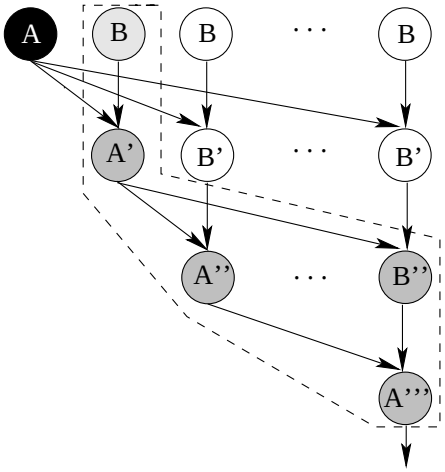


Figure 3.8: *Ejemplo de árbol de bloqueos en el flujo de trabajos de cd-hit.*

generado por un nodo integrante del camino crítico. $ST(n)$ puede calcularse de la siguiente manera:

$$ST(n) = \frac{n(1+n)}{2} \tag{3.4}$$

siendo la suma de los términos pertenecientes a una progresión aritmética.

La optimización finalmente empleada fue la replicación de todos los nodos del camino crítico. En particular, se crearon 3 copias de cada tarea.

3.7.2. Resultados Experimentales

Al probar la heurística escogida sobre el flujo de trabajos, se optó por procesar una base de datos más ambiciosa, tal y como se explica en los artículos incluídos en los Apéndices A.8 y A.9. Esta base de datos estaba compuesta por entradas de UniProt (RefSeq) y fragmentos de secuencia con origen en el meta-genoma del Mar de los Sargazos[5], todos distribuidos por el NCBI. Su tamaño era 1,7 GB y en ella estaban contenidas 4.186.285 proteínas. Por otro lado, las divisiones realizadas durante los experimentos fueron 32, 40 y 48.

La infraestructura Grid empleada en esta ocasión estaba compuesta por recursos de a su vez, dos infraestructuras a diferentes escalas: regional y mundial. Los recursos de computación pertenecientes a ambas están detallados en la Tabla 3.3.

[5]ftp://ftp.ncbi.nih.gov/genbank/wgs/

Table 3.3: *Recursos (EGEE y GRIDIMadrid) durante los experimentos con optimización.*

Centro	País	Procesador	Velocidad	Nodos
Recursos de GRIDIMadrid				
UCM	ES	P4	3216	2
CIEMAT	ES	P4	2392	22
Recursos de EGEE (BIOMED VO)				
BHAM-UNI	UK	PIII	800	128
BRUNEL	UK	P4	2000	5
CGG	FR	PIII	1266	56
CIEMAT	ES	PIII	1001	220
CYF-KR	PL	P4	2800	264
GRID-ACAD	BG	P4	2400	78
HELLASGRID	GR	P4	3400	356
IFCA	ES	P4	3200	96
II	MK	P4	3300	8
IMPERIAL	UK	P4	2000	188
IN2P3	FR	PIII	1001	569
INFN	IT	P4	2400	124
IPP-ACAD	BG	P4	2800	10
JET-EFDA	UK	PIII	1098	66
KELDYSH	RU	P4	3000	14
L.-HEP	UK	P4	3000	380
LIP	PT	P4	2200	52
MAN-UNI	UK	P4	2800	844
PNPI	RU	P4	3000	112
SAVBA	SK	P4	3200	41
SRCE	HR	P4	2193	16
UAM	ES	P4	2566	14
UCL	UK	P4	2800	312
UNI-LINZ	AT	P4	3014	8

Tanto las máquinas locales como regionales están cerca de aquella desde donde se enviaban los trabajos, y por lo tanto ofrecen una menor latencia. Por otro lado, las máquinas pertenecientes a la infraestructura del proyecto EGEE garantizan una exclusividad en el uso de sus CPUs, así como una mayor productividad y número, a pesar de la gran posibilidad de encontrar recursos ocupados. Una infraestructura

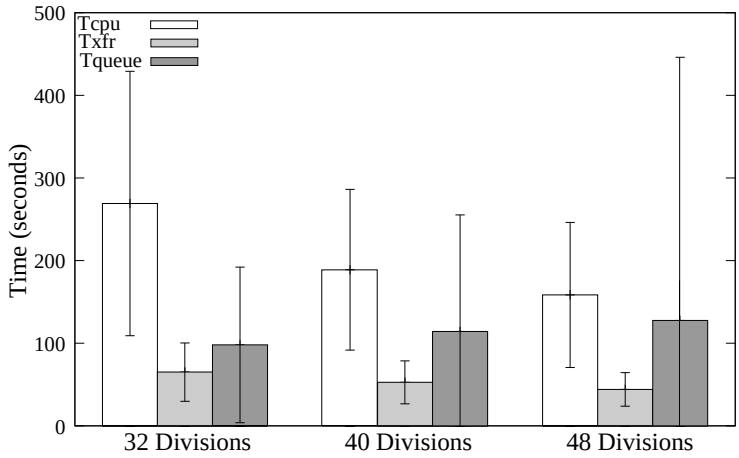


Figure 3.9: *Tiempos medios consolidados de CPU (T_{cpu}), transferencia (T_{xfr}) y espera en cola (T_{queue}) para cada número de particiones con y sin optimización.*

global de estas dimensiones se consideró necesaria debido al tamaño de la base de datos a procesar. Por tanto, la capacidad de GridWay para ofrecer un uso coordinado de sus sub-infraestructuras ya mencionada antes [VPHML06], fue determinante.

Se realizaron experimentos para cada número de divisiones considerado y, con el objetivo de evaluar la mejora, se realizaron experimentos adicionales sin replicación. Las tareas fueron lanzadas desde la Universidad Complutense de Madrid (UCM), que a su vez pertenece a GRIDIMadrid, en diferentes días de la semana durante Abril del 2007. Por otro lado, se redujo el número de tareas simultáneas a 10, para evitar saturar el ancho de banda debido al tamaño de los archivos transferidos.

La Figura 3.9 muestra los tiempos medios consolidados de CPU (T_{cpu}), transferencia (T_{xfr}) y espera en cola (T_{queue}), categorizados por número de divisiones realizadas sobre la base de datos de entrada. Tanto T_{cpu} como T_{queue} presentan una gran variabilidad. En el caso específico de T_{cpu}, esto fue debido a que recursos con diferente capacidad de computación estuvieron disponibles durante cada experimento. Por otro lado, la variabilidad de los sistemas de gestión de recursos locales pertenecientes a los recursos incidieron dramáticamente sobre T_{queue}. También T_{xfr} presenta variabilidad, debida sobre todo a los diferentes enlaces implicados.

La Figura 3.10 muestra los valores para diferentes tiempos de ejecución. T es el tiempo experimental del flujo de trabajos y T_{exp} es el *tiempo total esperado*. Este *tiempo total esperado* toma en cuenta exclusivamente las tareas pertenecientes al *camino crítico* obviando los fallos producidos en los trabajos. El valor de T_{exp} es entonces el

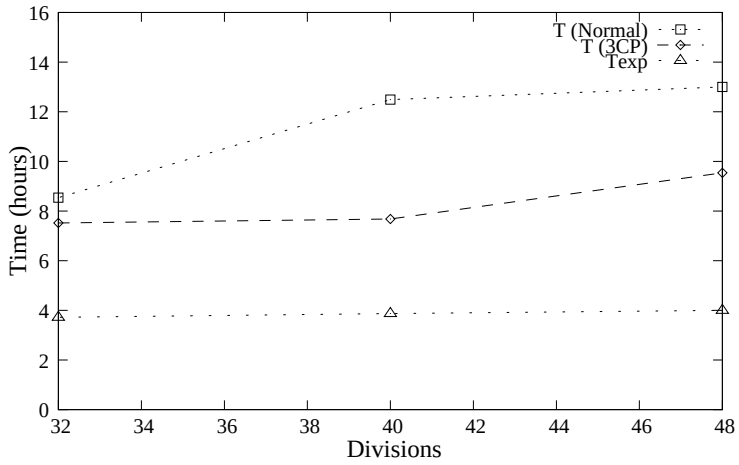


Figure 3.10: *Tiempos de ejecución del flujo de trabajos usando la replicación.*

mismo tanto con optimización como sin ella, y puede ser calculado tal que:

$$T_{exp} = N \cdot (T_{cpu} + T_{xfr} + T_{queue}), \tag{3.5}$$

donde N es el número de divisiones de la base de datos. Lo que se observa en la Figura 3.10 es que los tiempos experimentales aplicando la optimización son inferiores al resto.

El *speed-up* puede calcularse dividiendo el tiempo total de ejecución por el *tiempo secuencial* del flujo de trabajos (T_{seq}). Se puede estimar (T_{seq}) con la expresión:

$$T_{seq} = N \cdot T^A_{cpu} + T^B_{cpu} \cdot \frac{N \cdot (N+1)}{2}, \tag{3.6}$$

donde T^A_{cpu} y T^B_{cpu} son el tiempo de CPU de las tareas sombreadas y blancas de la Figura 3.7. Sus valores corresponden con los tiempos medios de ejecución que se obtuvieron durante los experimentos. En cualquier caso, conviene recalcar que no es posible realizar el proceso secuencial de la propia aplicación *cd-hit* en una nica CPU, siempre debido a las restricciones de memoria.

La Figura 3.11 muestra diferentes estimaciones del *speed-up*: el *speed-up experimental* (S) y el *speed-up* obtenido con el *tiempo total estimado* (S_{exp}). Se puede observar que la optimización permite aumentar el nivel de paralelismo, lo cual no sucede con caso contrario, donde S decrece con el número de las divisiones realizadas sobre la base de datos. No obstante, algunos aspectos tienen que ser considerados en este estudio. Primeramente, se redujo a 20 el número máximo de procesadores usados

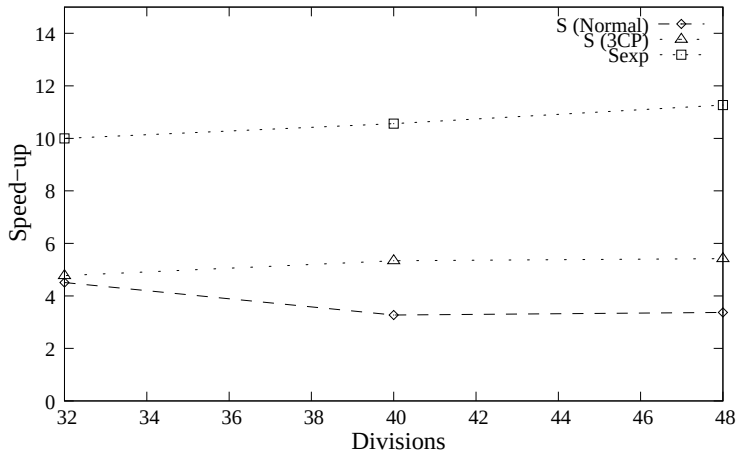


Figure 3.11: *Speed-up de la ejecución del flujo de trabajos con optimización y sin ella.*

simultáneamente debido a restricciones en la planificación. Por otro lado, el nivel de paralelismo de la aplicación siempre decrece en cada nivel debido a la forma del flujo de trabajos (*árbol de entrada*), alcanzando S_{exp} un valor máximo de 12,20 con 48 divisiones en la base de datos.

A pesar de los factores explicados anteriormente, la optimización representó un 50 % de la ganancia de *speed-up*. El valor 20 para el *speed-up* no pudo ser alcanzado debido a la naturaleza inherente al Grid, representada por el número de replanificaciones (mostradas en la Tabla 3.4). Adicionalmente, esta mejora se vio reflejada en los tiempos medios empleados para completar cada nivel, mostrados en la misma Tabla.

Table 3.4: *Número de trabajos replanificados y tiempos medios para cada experimento con y si optimización (3CP y Normal, respectivamente). Para los tiempos medios se consideró T_{cpu}, T_{xfr}, T_{queue} y las propias replanificaciones para cada nivel completado del algoritmo.*

Divisiones	Replanificaciones		Tiempo Medio	
	Normal	3CP	Normal	3CP
32	69	34	16,3′	14,7′
40	170	35	14,6′	11,7′
48	68	27	14,8′	12,1′

A pesar de que las heurísticas de replicación dotan de eficiencia a la ejecución

del flujo de trabajos, se debería cosiderar su coste. Dependiendo del número de divisiones realizadas sobre la base de datos, se pueden encontrar diferentes tiempos medios de ejecución y transferencia asociados a las dos tareas *descartadas* (aquellas tareas *replicadas* que son terminadas o terminan por su cuenta antes de que el sistema decida mandarles la señal correspondiente). Estos costes, tanto en la transferencia como en la ejecución, están detallados en la Tabla 3.5.

Table 3.5: *Costes de la replicación por trabajo y división de la base de datos.*

Divisiones	Tiempo Medio	
	T_{xfr}	T_{cpu}
32	45″	5′16″
40	30″	3′51″
48	25″	2′55″

Consecuentemente, se puede definir el coste total de la replicación como la suma de los tiempos antes mencionados. El coste para 32 divisiones fue 8 horas y 6 minutos. Para 40 divisiones, 7 horas y 25 minutos. Finalmente, para 48 divisiones fue 7 horas y 55 minutos. No obstante, se podría considerar el peor caso para las tareas *descartadas*, que consistiría en su ejecución en el recurso más lento y empleando el enlace con el ancho de banda más bajo para las transferencias. El objetivo de esto es estimar el límite superior del coste de la replicación. Para 32 divisiones sería 32 horas, 28 horas para 40 divisiones y finalmente, 49 horas para 48 divisiones.

3.7.3. Añadiendo la Aglomeración

La heurística de aglomeración también fue implantada en el sistema. Como se vió en la Sección 3.6.2, esta heurística consiste en ejecutar localmente las tareas pertenecientes a los últimos niveles del flujo de trabajos, con el fin de disminuir el tiempo total de transferencia y el de espera en cola.

Los experimentos se realizaron con la misma base de datos procesada anteriormente. En este caso, se practicaron 20 divisiones y en todos los experimentos se aplicó adicionalmente la heurística de replicación, creando una copia de cada tarea del *camino crítico*. En la Figura 3.12 se muestran los resultados de aglomerar las tareas de los niveles 16, 17 y 18, comparándolos con el tiempo empleado por el flujo de trabajos sin ninguna optimización. Como puede apreciarse, sólo es a partir de aplicar la aglomeración a partir del nivel 18 (penúltimo) que se consigue una mejora, ahorrando tiempos de transferencia y espera en cola. En los casos anteriores, el

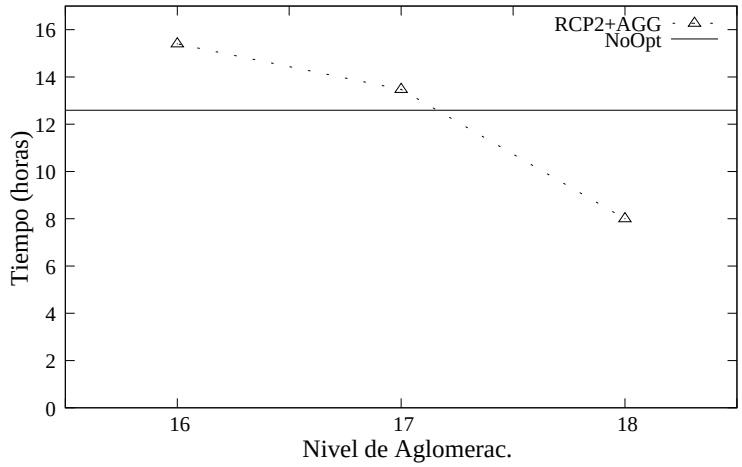


Figure 3.12: *Tiempos del flujo de trabajos aplicando replicación (2 tareas del camino crítico) y diferentes niveles de aglomeración ($RCP2 + AGG$), frente al de la ejecución sin optimización ($NoOpt$).*

sacrificio del nivel de paralelismo afecta en el tiempo total de ejecución.

3.8. Modelo de Ejecución del Flujo de Trabajos

Los esfuerzos para optimizar la ejecución de *cd-hit* descritos anteriormente pueden calificarse de prueba de concepto. La replicación es viable y constituye una mejora del *speed-up*. La aglomeración se presenta como un complemento para obtener un porcentaje adicional de optimzación. No obstante, ni el número de copias por tarea ni el nivel de aglomeración son valores que debieran dejarse al azar, mucho menos el número de divisiones realizadas sobre la base de datos.

Si bien existen diversos factores que no pueden predecirse, tales como el estado de los recursos de computación o la saturación de las comunicaciones, sí es posible producir un modelo de ejecución que permita acercarse al comportamiento real del flujo de trabajos. A través de este modelo se podría estudiar el impacto de los diferentes aspectos implicados y tomar así una decisión beneficiosa de antemano.

3.8.1. Modelo Básico

Con el fin de obtener la ecuación general que caracteriza la ejecución del flujo de trabajos, se realizaron varios experimentos sin optimización sobre la última base

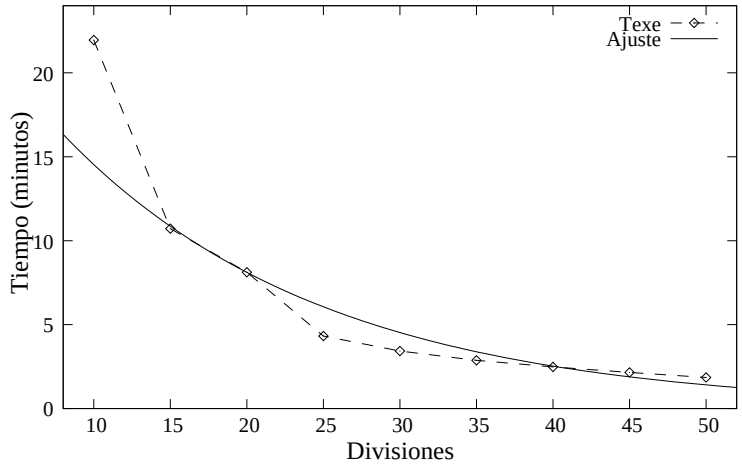


Figure 3.13: *Tiempos de ejecución del flujo de trabajos según número de divisiones y su curva de ajuste.*

de datos procesada y con un número diferente de divisiones. Para medir el tiempo de ejecución medio de las tareas pertenecientes al *camino crítico* (aquellas que determinan la ejecución de todo el flujo de trabajos), se extrajeron unas tareas tipo y se ejecutaron en una máquina local, un Pentium IV a 3,2 GHz. También de estas ejecuciones se calculó el tamaño de los archivos implicados en el proceso, con el fin de determinar posteriormente el tiempo de transferencia en base a la velocidad media de los enlaces. Ambas medidas se detallan en la Tabla 3.6.

Table 3.6: *Tiempos de ejecución (T_{exe}) de las tareas del camino crítico y tamaño de los archivos.*

Divisiones	T_{exe}	Tamaño (MB)
10	21,95′	509
15	10,72′	339
20	8,12′	254
25	4,32′	204
30	3,43′	170
35	2,87′	145
40	2,48′	127
45	2,15′	112
50	1,85′	101

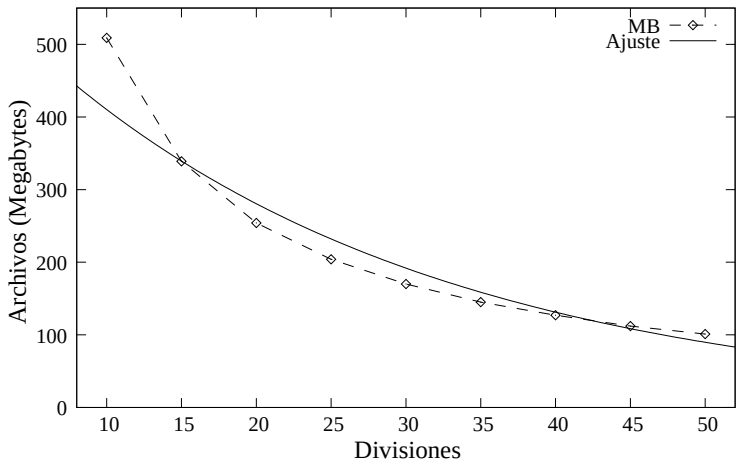


Figure 3.14: *Tamaño de los archivos del flujo de trabajos según número de divisiones y su curva de ajuste.*

El paso siguiente consistió en realizar un ajuste exponencial sobre los valores experimentales. Así, la ecuación que caracteriza el tiempo de ejecución (T_{exe}) de cada tarea perteneciente al *camino crítico* se expresa tal que:

$$T_{exe} = 26,038e^{-0,0583d}, \tag{3.7}$$

siendo d el número de divisiones de la base de datos. Tanto los valores experimentales como su curva de ajuste están representados en la Figura 3.13.

Por otro lado, el tiempo de transferencia (T_{xfr}) de los archivos asociados a dicha tarea:

$$T_{xfr} = \frac{599,96e^{-0,038d}}{60v}, \tag{3.8}$$

donde v corresponde al ancho de banda medio de los enlaces considerados en la infraestructura, medidos en MB/s. La Figura 3.14 representa los valores experimentales empleados y su correspondiente curva de ajuste.

Finalmente, se pueden relacionar estas dos ecuaciones con el fin de caracterizar el tiempo total de ejecución de todo el flujo de trabajos:

$$T = (d-1)(26,038e^{-0,0583d} + \frac{599,96e^{-0,038d}}{60v} + T_{queue}), \tag{3.9}$$

siendo el tiempo de espera en cola (T_{queue}) constante y calculándose con una media de los valores empíricos, puesto que dicho tiempo es independiente de las divisiones realizadas.

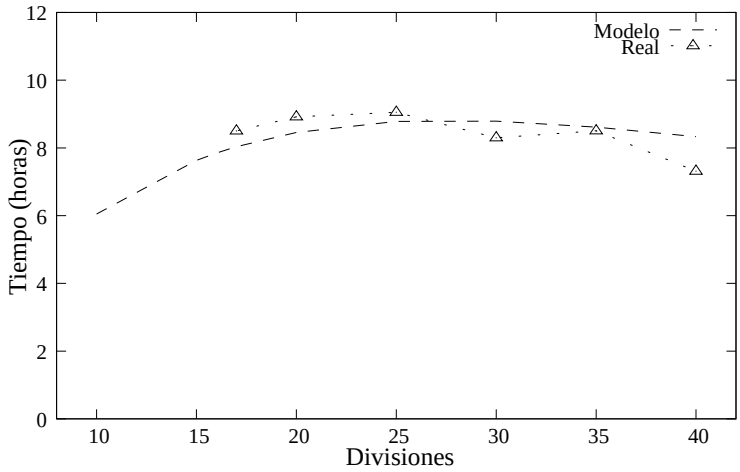


Figure 3.15: *Resultados reales de la ejecución del flujo de trabajos frente a las previstas por el modelo.*

Al planteamiento del modelo le siguió una demostración con resultados reales. Se fijó el tiempo de espera en cola a 3 minutos, en base a la media obtenida experimentalmente. Con el mismo procedimiento, se tomó 300 KB/s como velocidad media de los enlaces empleados. La Figura 3.15, que representa los valores estimados y los obtenidos empíricamente, demuestra la bondad del modelo. Hay que recalcar que no se hicieron pruebas con un número de divisiones inferior a 17, puesto que el tamaño alcanzado por los archivos de entrada era demasiado grande para ser procesados en los recursos remotos y en algunos casos, el tiempo de proceso excedía el marcado por las políticas del sitio.

3.8.2. Modelo con Optimizaciones

Una vez obtenido un modelo válido que caracterice la ejecución del flujo de trabajos, se proporcionarán a continuación, modelos (o variantes) que consideren las optimizaciones escogidas.

Replicación

Recordando lo explicado en la Sección 3.6.2, la replicación consiste en crear tareas suplementarias para ciertos nodos del flujo de trabajos, persiguiendo aumentar de esta forma las posibilidades de una rápida ejecución.

De esta forma, esta variante del modelo no persigue calcular un valor fijo para el

tiempo de ejeución, sino la densidad de tiempos para un grado de replicación, en un intervalo determinado y con un grado de confianza fijado. Al hablar de probabilidades, se debe primero averiguar la que corresponde a que cualquier tarea del flujo de trabajos tarde un tiempo determinado. Así, se debe realizar un análisis estadístico para los tiempos de ejecución, como el reflejado en la Tabla 3.7, en el que se usaron los resultados experimentales con 20 divisiones en la base de datos [IOP99].

Table 3.7: *Análisis estadístico de los tiempos de ejecución experimentales para 20 divisiones.*

Media (μ)	1544,11s
Desviación Típica (σ)	776,319s
Mediana	1341s
Curtosis Tipificada	-0,10485

El primer paso consiste en simular la replicación de una tarea dada perteneciente al *camino crítico*. Para ello se aplicará el **Teorema Central del Límite** [Par08], considerando como condiciones suficientes el que los tiempos de cada tarea replicada son variables independientes, están idénticamente distribuidas y tanto sus valores como la varianza son finitas. Posteriormente se fijará un valor v para el cual se pregunte la probabilidad de que el tiempo de una tarea tipo la supere. Dicho valor sirve como indicador del tiempo mínimo obtenido por el mejor caso de la replicación, y se puede construir multiplicando el grado de replicación por el límite superior del primer cuartil de los resultados experimentales.

Como ilustración se puede partir del siguiente ejemplo: *¿Con qué probabilidad el tiempo de ejecución del flujo de trabajos, habiendo realizado* 20 *divisiones, será menor de* 4 *horas, aplicando un grado de replicación* 2*?* Se fijaría v a 750, puesto que ese es el valor de dividir el tiempo total de ejecución por el número de niveles. A continuación, se empleará v en la siguiente ecuación, tal y como enuncia el **Teorema Central del Límite**:

$$Y = \frac{r(v-\mu)}{\sqrt{r\sigma^2}}, \tag{3.10}$$

siendo r el grado de replicación aplicado (2 en el ejemplo) y proviniendo μ y σ de un estudio estadístico inicial, como el reflejado en la Tabla 3.7 ($1544,11$ y $776,319$, respectivamente). Para el ejemplo citado, el valor de Y es $-1{,}44$. Posteriormente, se pasará a expresar el problema como:

$$W = P(Z < Y), \tag{3.11}$$

de tal forma que ya se estará trabajando con una distribución $N(0,1)$ como la representada en la Figura 3.16. Tras buscar el valor de W sustituyendo Y en las tablas de

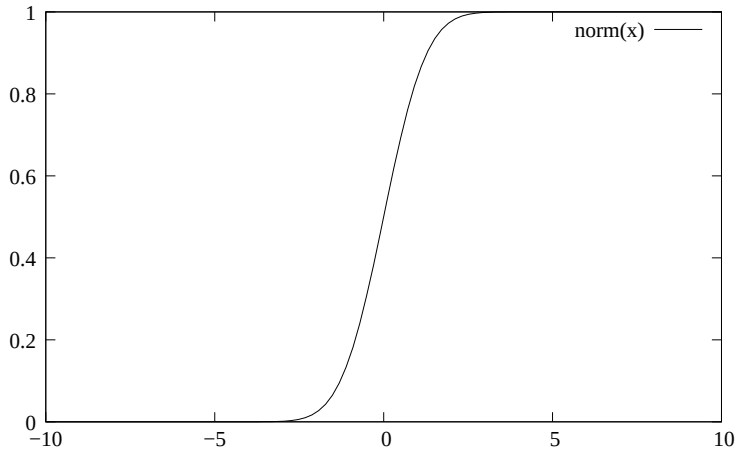


Figure 3.16: *Distibuión normal estándar.*

la distribución, se obtendrá la probabilidad buscada. Volviendo al ejemplo inicial, el valor de W es $0,07493$, con lo que se puede afirmar que el tiempo de ejecución del flujo de trabajos es menor de 4 horas con una probabilidad del $7,5\,\%$.

El uso de la replicación conlleva también unos costes, tal y como se observó al final de la Sección 3.7.2. Así, es posible calcular de forma sencilla el número total de tareas desaprovechadas (C_t) cuando se aplica esta mejora en el *camino crítico*:

$$C_t = (d-1)(r-1), \tag{3.12}$$

siendo d el número de divisiones y r, como en la anterior expresión, el grado de replicación. En el ejemplo, este valor sería 19. Pero el coste también puede ser expresado como el tiempo total empleado por todas las tareas descartadas. Para ello, se repitirá el mismo procedimiento empleado para obtener la probabilidad de una ejecución temprana de las tareas cambiando algunas condiciones de partida. En este caso, v correspondería a un valor alto, como por ejemplo, el límite inferior del tercer cuartil. Además, r en la Ecuación 3.10 se sustituiría por $(r-1)$, que corresponde al número de tareas descartadas. Finalmente, al preguntar por la probabilidad de que el tiempo supere ese valor dado, la expresión quedaría tal que:

$$W = 1 - P(Z > Y), \tag{3.13}$$

con el fin de emplear las tablas de la distribución normal.

Aglomeración

La modelización de esta heurística, que consiste en ejecutar localmente las tareas pertenecientes a los últimos niveles del flujo de trabajos tal y como se explica en la Sección 3.6.2, se reduce a calcular el tiempo secuencial de las tareas implicadas tal que:

$$T_{agg} = \sum_{i=0}^{d-n-1} (d-n-i)T_{exe}, \tag{3.14}$$

donde d es el número de divisiones realizadas sobre la base de datos y T_{exe}, el tiempo de ejecución estimado por la Ecuación 3.7 del modelo sin optimizaciones. Por su parte, n corresponde al nivel escogido, considerando el valor 1 para el primero que se ejecuta en el flujo de trabajos. Suponiendo un ejemplo en el que se practican 35 divisiones a una base de datos (d) y se escoge aglomerar desde el antepenúltimo nivel, el número 32 (n), T_{agg} resulta 28 minutos.

Al suprimir las tareas pertenecientes a los últimos valores, el modelo de ejecución en el Grid cambia. Así, la ecuación que lo caracteriza queda de la siguiente manera:

$$T_{grid} = (n-1)(T_{exe} + T_{xfr} + T_{queue}), \tag{3.15}$$

siendo n el nivel escogido para la aglomeración. Tanto T_{exe} como T_{xfr} se calculan mediante las Ecuaciones 3.7 y 3.8, respectivamente. Por otro lado, el valor de T_{queue} es fijo, tal y como se explica en la Sección 3.8.1. En el ejemplo antes citado, considerando $T_{queue} = 3$ minutos y $v = 0,3$ MB/s, se obtiene que T_{grid} es 496 minutos.

Finalmente, si se unen los dos tiempos calculados anteriormente en una expresión:

$$T = (n-1)(T_{exe} + T_{xfr} + T_{queue}) + \sum_{i=0}^{d-n-1} (d-n-i)T_{exe}, \tag{3.16}$$

se obtendrá el tiempo total de ejecución del flujo de trabajos, aplicando la aglomeración a partir del nivel deseado. Para finalizar el ejemplo propuesto, el tiempo total de ejecución será 526 minutos ($8,7$ horas), frente a los 544 minutos ($9,1$ horas) sin aplicar esta mejora.

3.9. Conclusiones

Las necesidades planteadas al Grid han ido incrementando su complejidad. Con el devenir de los años, las aplicaciones ya no sólo requerían que un número ingente de tareas se ejecutaran a la vez, sino que además existieran relaciones de dependencia entre ellas. El Ser Humano, en su afán por modelizar los problemas de

la mundo real, concibió los flujos de trabajo, cuyo Estado del Arte ha sido expuesto en este Capítulo.

Así, la Investigación se ha centrado en un flujo de trabajos particular, perteneciente a un área científica emergente, la Bioinformática. Tras estudiar diferentes heurísticas de optimización pertenecientes a la Literatura y también mostrar su Estado del Arte, se escogió la replicación y la aglomeración, validando los resultados experimentalmente.

Una vez demostrada la viablidad tanto del portado de la aplicación como de las optimizaciones, se ha procedido a confeccionar un modelo de ejecución de todo el sistema. Su objetivo es el conocer a prori el resultado de aplicar ciertos valores a parámetros determinantes, con el fin de optimizar la ejecución. Así, se ha presentado un modelo básico que permite predecir el tiempo de ejecución y transferencia en base al número de divisiones de la base de datos. Tras validarlo con resultados experimentales, se han desarrollado dos modificaciones del mismo que permiten además, predecir el efecto de las mejoras estudiadas.

Uno de los resultados ha sido el portado de *cd-hit*, una apliación extensamente utilizada por la comunidad bioinformática. Este portado al Grid se hacía imprescindible, puesto que los límites de memoria de las máquinas convencionales hacían imposible procesar unas bases de datos cada vez más grandes, y en el mejor de los casos, el proceso acababa eternizándose. El trabajo perteneciente a este Capítulo no se ha limitado al portado, sino que se ha provisto al sistema de las dos heurísticas escogidas para optimizar la ejecución. Este sistema final, programado empleando el estándar DRMAA del Open Grid Forum y empleando el metaplanificador GridWay, se distribuye con licencia de código abierto a toda la comunidad científica. Además, el CNIO, institución que propuso el portado de *cd-hit*, la está empleando con bases de datos de producción.

Capítulo 4

Principales Aportaciones y Trabajo Futuro

> Los límites sólo existen en tu imaginación.
>
> Luis Vázquez

4.1. Principales Aportaciones

Tras mostrar el estado del arte de la Computación Grid, la presente Tesis se ha centrado en el estudio de dos perfiles de aplicación sobre los que muchos problemas del mundo real se reflejan. A lo que Grid respecta, se han introducidos sus conceptos y elementos principales, así como un conjunto de proyectos e infraestructuras más relevantes a modo de ejemplo.

Una infraestructura Grid no tendría razón de existir sin aplicaciones que la utilicen. La Investigación de la presente Tesis de Doctorado se ha centrado en un perfil de aplicaciones particular: los flujos de trabajos. Así, se ha comenzado analizando las aplicaciones de *barrido de parámetros*, que es el ejemplo más sencillo, pero cuya ejecución eficiente es un desafío debido a las condiciones que caracterizan una infraestructura Grid. Posteriormente, se ha estudiado un *árbol de entrada*, de gran complejidad. En ambos casos, tras describir su Estado del Arte, se ha procedido a estudiar heurísticas de optimización para mejorar su eficiencia. Para la primera aplicación se emplearon *chunks* y para las segunda, *replicación* y *aglomeración*. El trabajo desarrollado a este punto ha consistido en dotar a los algoritmos originales de una extensión que les permita beneficiarse de las mejoras otorgadas por estas heurísticas. Los sistemas finales han sido validados con datos reales, demostrando su idoneidad y eficiencia.

También se ha dotado al flujo de trabajos de la aplicación bioinformática (la más compleja) de un modelo de ejecución. Dicho modelo permite identificar las condiciones de partida idóneas, esto es, el número de divisiones practicadas sobre la base de datos procesada. En una iteración posterior, se extrajeron también los modelos resultantes de aplicar las optimizaciones escogidas.

El estudio de los estos flujos de trabajos se ha realizado a través de aplicaciones reales, pertenecientes a áreas científicas emergentes (Física de Fusión y Bioinformática). Estas aplicaciones no solo son de gran demanda para las instituciones que solicitaron su portado (CIEMAT y CNIO), sino también para el resto de la comunidad científica. En ambos caso, se procedió a portarlas al Grid, sirviendo los sistemas finales como trampolín para el estudio de las optimizaciones propuestas. El resultado final no se ha limitado a portar las respectivas aplicaciones *gridificadas* de forma eficaz, pues con la aplicación de las heurísticas, se ha dotado de eficiencia a ambos sistemas.

Los sistemas finales fueron creados empleando estándares del Open Grid Forum y son distribuidos con licencia de código abierto, con el objetivo de estar al alcance de toda la comunidad científica. Además, tanto CIEMAT como CNIO los están empleando con datos reales y en entornos de producción.

4.2. Trabajo Futuro

Considerando la complejidad inherente a los flujos de trabajo y en particular a la aplicación bioinformática tratada, su optimización se presenta como un campo en el que quede bastante por explorar. Así se aplicó sólo una variante de la *replicación*, quedando muchas por explorar y por evaluar. A lo que los modelos formulados para estas optimizaciones respecta, quedan pendientes muchas más pruebas que corroboren su viabilidad.

Obviamente, a la implantación y estudio del mayor número de optimizaciones seguirá la formulación de modelos válidos que faciliten su elección antes de procesar las ya ingentes bases de datos de proteínas. Con el conocimiento adquirido hasta este punto, se estará en grado de estudiar otros flujos de trabajos pertenecientes a aplicaciones cuya relevancia científica y demanda de Computación Grid sean notables.

Capítulo 5

Principali Contribuzioni e Futuri Sviluppi del Lavoro

> I limiti solo esistono nella tua immaginazione.
>
> Luis Vázquez

5.1. Principali Contribuzioni

Dopo aver esposto una panoramica della situazione attuale della Computazione Grid, questa tesi si è centrata sullo studio di due profili di applicazione nei quali si rispecchiano molti problemi del mondo reale. Con particolare riferimento a Grid, ne sono stati esaminati i concetti e gli elementi principali, così come sono stati introdotti, a modo d´ esempio, i progetti e le infrastrutture più rilevanti. Una infrastruttura Grid non avrebbe ragione di esistere senza applicazioni che la usino. La Ricerca della presente Tesi di Dottorato è centrata su un particolare profilo di applicazioni: i workflow. Si è iniziato dall´ analisi delle applicazioni di *parameter sweep*, che è l´ esempio più semplice, per passare poi a studiare un intree di grande complessità. In entrambi i casi, dopo averne descritto lo Stato dell´ Arte, si è proceduto a studiare euristiche di ottimizzazione per migliorare la loro efficienza. Per la prima applicazione sono stati utilizzati dei chunks e per la seconda, repplicazione ed agglomerazione. A questo punto il lavoro è consistito nel dotare gli algoritmi originali di un´ estensione che permettesse loro beneficiarsi dei miglioramenti ottenuti per mezzo di queste euristiche. I sistemi finali sono stati validati con dati reali e hanno dimostrato la loro idoneità ed efficienza. Il workflow dell´ aplicazione bioinformatica (la più complessa) si è dotato anche di un modello di esecuzione. Questo modello permette di identificare le condizioni di partenza idonee, tali come il numero di divisioni fatte alla base dati processata. In una iterazione posteriore, sono stati prodotti anche modelli ottenuti applicando le ottimizzazioni scelte.

Lo studio di questi workflow si è realizzato mediante applicazioni reali che appartengono ad aree scientifiche emergenti (Fisica di Fusione e Bioinformatica). Ques-

te applicazioni non sono richieste soltanto dagli Enti che hanno sollecitato il loro porting (CIEMAT e CNIO), ma anche dal resto della comunità scientifica. In entrambi i casi, si è proceduto al porting usando i sistemi finali per studiare le ottimizzazioni proposte. Il risultato finale non si è limitato a fare porting delle applicazioni in modo efficace, ma con l´ implementazione delle euristiche, entrambi i sistemi sono stati dotati di efficienza.

I sistemi finali sono stati creati adoperando gli standard dell´ Open Grid Forum e sono distribuiti con licenza Open Source, con l´ obbiettivo di arrivare a tutta la comunità scientifica. Inoltre, CIEMAT e CNIO li stanno usando con dati reali in infrastrutture di produzione.

5.2. Futuri Sviluppi

Data la complessità inerente ai workflows e in particolare, all´ applicazione bioinformatica studiata, la loro ottimizzazione si presenta come un vasto campo dove c´ è ancora tanto da esplorare. Qui si è lavorato soltanto con una variante, ma sono tante quelle che non sono state ancora verificate e valutate.

Ovviamente, come conseguenza di aver implementato e studiato il maggior numero di ottimizzazioni, verrà la formulazione di modelli che ne facilitino la scelta prima di processare le già ingenti basi di dati di proteine. A questo punto, con le conoscenze acquisite, si sarà in grado di studiare altri workflows appartenenti ad applicazioni con notevole attinenza scientifica e rilevante necessità di Computazione Grid.

Capítulo 6

Principal Contributions and Future Work

> Limits only exist in your imagination.
>
> Luis Vázquez

6.1. Principal Contributions

After showing Grid Computing´ s State of the Art, the present PhD. Thesis had focused on the study of two application profiles which do reflect many real world problems. Regarding Grid Computing, its concepts and main elements have been introduced, as well as the most relevant projects and infrastructures.

A Grid infrastructure looses its reason to be without applications that use it. This Research focuses on a particular application profile: workflows. This way, *parameter sweep* applications, the simplest ones, were analyzed at first. Even being simple, their efficient execution is challenging due to conditions characterizing a Grid infrastructure. Then, an complex *intree* workflow was studied. In both cases and after describing their State of the Art, Research went on with optimization heuristics, in order to increase efficiency. In the first application, *chunks* were employed, and in the second one, *replication* y *agglomeration*. The next step was to provide the original algorithms with an extension that would allow them to take advantage of these heuristics. Final systems have been validated with real data, demonstrating their suitability and efficiency.

Also, an execution model have been provided for the bioinformatics workflow (the most complex one). This model allows to identify the best start conditions, hence the input database divisions. In a further iteration, models considering the optimizations were also introduced.

The study of these workflows was done through real applications, pertaining to emergent scientific areas (Fusion Physics and Bioinformatics). These applications are not only highly demanded by the institutions who proposed their porting (CIEMAT and CNIO), but also by the rest of the scientific community. The final result was

not only their *porting* onto the Grid, also efficiency was conferred to them thanks to heuristics.

The final systems where created using Open Grid Forum standards and are distributed under a open source license, in order to reach the whole scientific community. Additionally, CIEMAT and CNIO are using them with real data in production environments.

6.2. Future Work

Considering the inherent complexity to workflows and in particular, to the bioinformatics applications, its optimization is presented as a wide are where there's much to explore and evaluate. Regarding the models that have been introduced for these optimizations, more tests are pending in order to demonstrate their suitability.

Obviously, implementing and studying the more optimizations will follow the formulation of valid models which will allow their selection before processing protein databases. With this knowledge achieved, study of other workflows pertaining to other interesting applications, with notorious scientific relevance and demand of Grid Computing, will be possible.

Apéndice A

Artículos Presentados

Nanos gigantium humeris
insidentes.

Bernardus Carnotensis

Stand on the shoulders of giants.

Google Scholar

A.1. Coordinated Harnessing of the IRISGrid and EGEE Testbeds with GridWay

Cita Completa

J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *Coordinated Harnessing of the IRISGrid and EGEE Testbeds with GridWay*. Journal of Parallel and Distributed Computing, vol. 66, n. 5 (IPDPS'04 Special Issue), Mayo 2006, pp. 763-771.

Abstract

Since the late 1990s, we have witnessed an extraordinary development of Grid technologies. Nowadays, different Grid infrastructures are being deployed within the context of growing national and transnational research projects. However, the coexistence of those different infrastructures opens an interesting debate about the coordinated harnessing of their resources, from the end-user perspective, and the simultaneous sharing of resources, from the resource owner perspective. In this paper we demonstrate the efficient and simultaneous use of different Grid infrastructures through a decentralized and *end-to-end* scheduling and execution system. In particular, we evaluate the coordinated use of the EGEE and IRISGrid testbeds in the

execution of a Bioinformatics application. Results show the feasibility of building *loosely-coupled* Grid environments only based on Globus services, while obtaining non trivial levels of quality of service, in terms of performance and reliability. Such approach allows a straightforward resource sharing since the resources are accessed by using *de facto* standard protocols and interfaces.

Referencia de Citas Bibliográficas

Alfieri et al. (2003); Allan et al. (2003); Baker et al. (2002); Bastolla et al. (2000); Carpenter (1996); Czajkowski et al. (1998); Delgado et al. (2004); Foster (2002); Foster y Kesselmann (1997); Huedo et al. (2004); José et al. (2003); Merzky et al. (2005); Pinedo (2002); Rajic et al. (2003); Schopf (2001); Schopf et al. (2002); Smirnova et al. (2003); van Ham (2003)

Índice de Impacto

0,43 en 2006, tal y como recoge el Journal Citation Reports 2007 (Tohmson Scientific).

Available online at www.sciencedirect.com
SCIENCE DIRECT®
J. Parallel Distrib. Comput. 66 (2006) 763–771
Journal of Parallel and Distributed Computing
www.elsevier.com/locate/jpdc

Coordinated harnessing of the IRISGrid and EGEE testbeds with Grid*W*ay☆

J.L. Vázquez-Poletti[a], E. Huedo[b], R.S. Montero[a,*], I.M. Llorente[a,b]

[a]*Departamento de Arquitectura de Computadores y Automática, Facultad de Informática, Universidad Complutense de Madrid, 28040 Madrid, Spain*
[b]*Laboratorio de Computación Avanzada, Simulación y Aplicaciones Telemáticas, Centro de Astrobiología (CSIC-INTA), 28850 Torrejón de Ardoz, Spain*

Received 18 March 2005; received in revised form 12 September 2005; accepted 3 January 2006
Available online 14 February 2006

Abstract

Since the late 1990s, we have witnessed an extraordinary development of Grid technologies. Nowadays, different Grids are being deployed within the context of a growing number of national and transnational research projects. However, the coexistence of those different infrastructures involves two challenging issues, namely: (i) simultaneous and coordinated use of resources from different Grids, from the end user perspective; and (ii) the simultaneous contribution of resources to different Grids, from the resource owner perspective. In this paper, we demonstrate that a decentralized and "end-to-end" scheduling and execution system can efficiently interoperate different Grids. In particular, we evaluate the coordinated use of the EGEE and IRISGrid testbeds in the execution of a Bioinformatics application. Results show the feasibility of building *loosely coupled* computational Grid environments only based on Globus services, while obtaining non-trivial levels of quality of service, in terms of performance and reliability. Such approach allows a straightforward resource sharing since the resources are accessed by using de facto standard protocols and interfaces.

Keywords: Grid computing; Globus Toolkit; Bioinformatics

1. Introduction

Grids of several scales are being deployed within the context of different research projects, whose final goal is to provide large-scale, secure and reliable sharing of resources among virtual organizations. A Grid, as defined by Foster [10], is a system (i) not subject to a centralized control, and (ii) based on standard, open and general-purpose interfaces and protocols, while (iii) providing some level of quality of service (QoS), in terms of security, throughput, response time or the coordinated use of different resource types. However, from our point of view, it is debatable whether some of these projects embrace the Grid philosophy, and to what extent. There is a tendency to ignore the first two requirements in order to get higher levels of QoS, mainly performance and reliability, for a given application scope. However, these requirements are even more important because they are the key to the success of *the Grid*.

The high expectations raised by Grid computing have favored the development and deployment of a growing number Grid infrastructures (testbeds). However, the interoperability between these Grids—each one with its own middleware developments, adaptations or extensions—is, to some extent, very limited, so reducing the potential large-scale application of the Grid from the user and resource owner point of views. In order to interoperate resources from *different* Grids it is necessary to assure the compatibility of the middleware components (e.g. authorization or information services) of each Grid. In this way, Allan et al. [2] have previously proposed the deployment of ad hoc policies and configurations to make compatible the core middleware services. So, higher level tools like resource brokers can transparently operate across overlapping Grids.

Another approach, more in line with the Grid philosophy, is the development of client tools that can interface different middleware implementations. Therefore, if we consider that most of the current projects are based on the Globus Toolkit, it is theoretically possible to move functionality from resources to

☆This research was supported by Ministerio de Ciencia y Tecnología, through the research Grants TIC 2003-01321 and 2002-12422-E, and by Instituto Nacional de Técnica Aeroespacial "Esteban Terradas" (INTA)—Centro de Astrobiología. The authors participate in the EGEE project, funded by the European Union under contract IST-2003-508833.
* Corresponding author. Fax: +34 91 394 46 87.
E-mail address: rubensm@dacya.ucm.es (R.S. Montero).

0743-7315/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2006.01.002

brokers or clients. This will allow the resources to be accessed in a standard way, making the task of sharing resources between Grids easier.

In this work, we demonstrate the feasibility of the last approach to interoperate *existing* Grid infrastructures. In particular, we will concentrate on computational Grids, which are mainly used for running compute-intensive jobs that potentially process large data sets. As we will see, this approximation also enables the construction of computational Grid environments only based on Globus services and user-level middleware, while obtaining non-trivial levels of QoS.

To this end, we consider a testbed built up from resources inside IRISGrid, [1] the Spanish National Grid Initiative, and resources inside EGEE (Enabling Grids for E-sciencE), [2] the European production-level Grid infrastructure. We analyze the coordinated use of these testbeds with the execution of a Bioinformatics application, since this is one of the most resource-demanding fields.

The structure of the paper is as follows. In Section 2, we will discuss some aspects about Grid infrastructure and middleware. Section 3 reviews a non-exhaustive list of current middleware and infrastructure projects and their relation with the Grid philosophy. The resulting experimental testbed and the target application are described in Sections 4 and 5, respectively, while Section 6 presents the obtained results and evaluates the performance of the overall infrastructure. Finally, Section 7 ends up with some conclusions.

2. A loosely coupled overlap of grids

We will hereafter refer to the computational environments created by strictly adhering to the previous definition (Section 1) as *loosely coupled* Grids. Loosely coupled Grids resemble the architecture of the Internet which is based on the "end-to-end" principle, and has fostered the spectacular development of the Internet and Web technologies in the past decade [7]:

> The basic argument is that, as a first principle, certain required end-to-end functions can only be performed correctly by the end-systems themselves.

A Grid infrastructure is usually decomposed into the following layers [3]: Grid applications and portals; user-level Grid middleware; core Grid middleware; and Grid fabric. The two internal layers are called "the middleware", since they connect applications with resources, or Grid fabric. In a *loosely-coupled* Grid, it is important to keep these layers separated and independent with a limited and well defined set of interfaces and protocols between them.

The main characteristics of these environments are [3] autonomy (of the multiple administration domains), heterogeneity, scalability and dynamism. These properties completely determine the way that scheduling and execution on Grids have to be done. For example, scalability and autonomy prevent the deployment of centralized resource brokers with total control over client requests and resource status. On the other hand, the dynamic resource characteristics in terms of availability, capacity and cost make essential the ability to adapt job scheduling and execution to these conditions.

The Globus Toolkit [11] has become a de facto standard core middleware in Grid Computing. Globus services allow secure and transparent access to resources across multiple administrative domains, and serve as building blocks to implement the stages of Grid scheduling [24]. Globus architecture follows an hourglass approach, which is indeed an "end-to-end" principle. Therefore, instead of succumbing to the temptation of tailoring the core Grid middleware to our needs (since in such case the resulting infrastructure would be application specific), or homogenizing the underlying resources (since in such case the resulting infrastructure would be a highly distributed cluster), we propose to strictly follow the "end-to-end" principle. Clients should have access to a wide range of resources provided through a limited and standardized set of protocols and interfaces. In the Grid, these are provided by the core Grid middleware, i.e. Globus. Just as, in the Internet, they are provided through the TCP/IP set of protocols.

The application of the "end-to-end" principle to Grid computing requires user-level middleware in the client side to make it easier and more efficient the execution of applications. Such client middleware should provide the end user with portable programming paradigms and common interfaces, which are being standardized by the *Global Grid Forum* (GGF), see, for example, the Distributed Resource Management Application API (DRMAA) [23] and the Simplified API for Grid Application (SAGA) [21].

At the other end, resource management software is advisable in Grid fabric to provide system administrators with tools to determine the amount of resources they are willing to devote to the Grid, avoiding their saturation by Grid jobs. Such kind of software [18] will aid in the expansion of the Grid, leading resource owners to embrace Grid technologies and share their resources with more confidence.

Moreover, the "end-to-end" principle reduces the firewall configuration to a minimum, which is also a welcome advance for security administrators. The more confident resource owners are, the more nodes they will add to the Grid, overcoming the typical scenario where administrators share only a small fraction of their hosts due to their mistrust of the Grid. We should consider that the Grid not only involves the technical challenge of constructing and deploying this vast infrastructure, it also brings up other issues related to resource sharing and security policies. Undoubtedly, an approach that allows administrators to have full control of their resources could help to overcome these socio-political difficulties [25].

3. The Grid philosophy in some existing projects

The EGEE project is creating the largest production-level Grid infrastructure in the world, which provides a level of performance and reliability never achieved before. A very restrictive set of requirements has been established for organizations that wish to take part in it. EGEE defines the user-level Grid middleware, the core Grid middleware and the Grid fabric, and

[1] http://irisgrid.rediris.es
[2] http://www.eu-egee.org

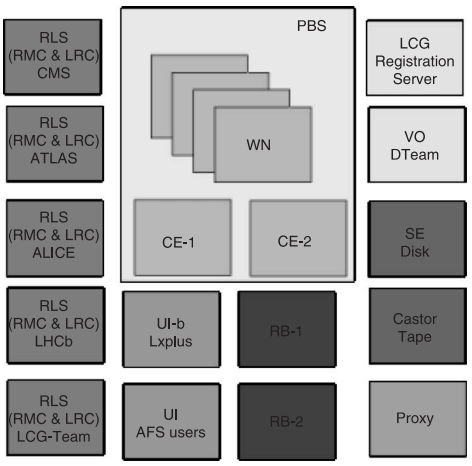


Fig. 1. LCG-2 available services at CERN (each service usually corresponds with a separate server node) [9].

so these layers are tightly related. EGEE uses the LCG (LHC Computing Grid) [3] middleware, LCG-2 (see Fig. 1).

This architecture presents some limitations in terms of heterogeneity, as it has a fixed configuration for clusters. The scalability of its deployment is also limited, as the middleware should be installed on the compute nodes, and they should have network connectivity. Also, LCG's focus is mainly on particle physics applications that depend on a single organization, CERN (the European Organization for Nuclear Research). Nevertheless, it is expected that the new EGEE middleware, gLite, [4] will overcome to some extent some of these limitations.

On the other hand, the main goal of the IRISGrid initiative is the creation of a stable national Grid infrastructure. The mission of IRISGrid is to establish the minimum protocols, procedures and guidelines for creating a research Grid in Spain. IRISGrid only defines the core Grid middleware within the Grid fabric, and it requires user-level Grid middleware to be fully functional. The first version of the IRISGrid testbed is based only on Globus, and it has been widely used through the Grid*Way* framework [16] (see Section 4.2).

An intermediate approach has been developed in other projects like NorduGrid, [5] whose architectural requirements are very close to those of IRISGrid. The requirement of being based only on Globus basic services is not considered. NorduGrid defines the user-level and the core Grid middleware, while leaving some flexibility in the Grid fabric [26]. However, it presents some interesting benefits like scalability and no single point of failure; the computational resources do not have to be exclusively dedicated to Grid, and have few site requirements (e.g. shared file system).

The NorduGrid user-level middleware uses an extended version of the Resource Specification Language (RSL) [8] and a smart front-end for job submission to a cluster, the Grid Manager (GM) [19]. The GM uses the GridFTP interface for job submission, instead of the "standard" Globus Resource Allocation Manage (GRAM) protocol. However, GM provides a virtual directory tree; access control based on the user certificate's DN; and access to meta-data catalogs. There are some ongoing efforts to provide interoperability between LCG and NorduGrid middlewares.

4. Overlap of the IRISGrid and EGEE testbeds

This work has been possible thanks to the collaboration of those research centers and universities that temporarily shared some of their resources. The final Grid environment results in a very heterogeneous infrastructure, since it presents several middlewares, architectures, processor speeds, resource managers (RM) and network links. In this section, we describe the main characteristics of the overlapped testbed, as well as the user-level middleware used in this research.

4.1. Grid fabric and core Grid middleware

Some of the centers that took part in the experiment collaborate on the Spanish Grid Initiative and Research Testbed (IRISGrid), which is composed of around 40 groups from different Spanish institutions. Seven of them participated in the experiment, donating a total of 195 CPUs. Details of the IRISGrid resources are shown in Table 1. Another 7 institutions from the EGEE project took part in the experiment contributing 333 CPUs, as shown in Table 2.

Together, the *overlapped* testbed is composed by 13 sites (note that LCASAT-CAB is both in IRISGrid and EGEE) and 528 CPUs. In the experiments below, the number of jobs simultaneously submitted to the same resource were limited to four, to not saturate the testbed, so only 64 CPUs were used at the same time. All sites are connected by RedIRIS, the Spanish Research and Academic Network. The geographical location and interconnection links of the different nodes are shown in Fig. 2.

The core Grid middleware components used in the experiments are shown in Table 3. We had to introduce some changes in the security infrastructure in order to perform the experiments. We used an user certificate issued by DATAGRID-ES CA for authentication, so IRISGrid sites had to give trust to this CA on their resources. Regarding authorization, we had to add an entry for the user in the `grid-mapfile` in both IRISGrid and EGEE resources.

Another possibility would be to use two different user certificates, each one to access each testbed. Moreover, in large projects it is common to deploy a VO management system, like VOMS [1], so it could be possible to create gateways

[3] http://lcg.web.cern.ch
[4] http://www.glite.org
[5] http://www.nordugrid.org

Table 1
IRISGrid resources contributed to the experiment

Name	Site	Network	Processor	Speed	Nodes	RM
heraclito	RedIRIS	Central	Intel Celeron	700 MHz	1	Fork
platon	RedIRIS	Central	2 × Intel PIII	1.4 GHz	1	Fork
descartes	RedIRIS	Central	Intel P4	2.6 GHz	1	Fork
socrates	RedIRIS	Central	Intel P4	2.6 GHz	1	Fork
aquila	DACYA-UCM	Madrid	Intel PIII	700 MHz	1	Fork
cepheus	DACYA-UCM	Madrid	Intel PIII	600 MHz	1	Fork
cygnus	DACYA-UCM	Madrid	Intel P4	2.5 GHz	1	Fork
hydrus	DACYA-UCM	Madrid	Intel P4	2.5 GHz	1	Fork
babieca	LCASAT-CAB	Madrid	Alpha EV67	450 MHz	30	PBS
bw	CESGA	Galicia	Intel P4	3.2 GHz	80	PBS
llucalcari	IMEDEA	Baleares	AMD Athlon	800 MHz	14	PBS
augusto	DIF-UM	Murcia	4 × Intel Xeon[a]	2.4 GHz	1	Fork
caligula	DIF-UM	Murcia	4 × Intel Xeon[a]	2.4 GHz	1	Fork
claudio	DIF-UM	Murcia	4 × Intel Xeon[a]	2.4 GHz	1	Fork
lxsrv1	BIFI-UNIZAR	Aragón	Intel P4	3.2 GHz	50	SGE

[a]These resources actually present two physical CPUs but they appear as four logical CPUs due to hyper-threading.

Table 2
EGEE resources contributed to the experiment

Name	Site	Network	Processor	Speed (GHz)	Nodes	RM
ce00	LCASAT-CAB	Madrid	Intel P4	2.8	8	PBS
mallarme	CNB	Madrid	2×Intel Xeon	2.0	8	PBS
lcg02	CIEMAT	Madrid	Intel P4	2.8	6	PBS
grid003	FT-UAM	Madrid	Intel P4	2.6	49	PBS
gtbcg12	IFCA	Cantabria	2×Intel PIII	1.3	34	PBS
lcg2ce	IFIC	Valencia	AMD Athlon	1.2	117	PBS
lcgce02	PIC	Cataluña	Intel P4	2.8	69	PBS

between them, so we can have a VO in EGEE consisting of all the IRISGrid users and vice versa [2].

4.2. User-level Grid middleware

As has been previously discussed, a loosely coupled overlap of computational Grids requires an user-level middleware in the client side, to perform the scheduling and execution management of jobs. In this work, we will use the Grid*W*ay framework [16]. Grid*W*ay is a light-weight meta-scheduler that performs job execution management and resource brokering. It allows unattended, reliable, and efficient execution of jobs, array jobs, or complex jobs on heterogeneous and dynamic Grids. Grid*W*ay performs all the job scheduling and submission steps transparently to the end user and adapts job execution to changing Grid conditions by:

- *Adaptive scheduling*: the Grid*W*ay framework periodically adapts the schedule to the available resources and their dynamic characteristics [16]. Grid*W*ay incorporates a *resource selector* that reflects the applications demands, in terms of requirements and preferences, and the dynamic characteristics of Grid resources, in terms of load, availability and proximity (bandwidth and latency).
- *Adaptive execution*: The Grid*W*ay framework also provides adaptive job execution to migrate running applications to more suitable resources, improving application performance by adapting it to the dynamic availability, capacity and cost of Grid resources. Moreover, an application can migrate to a new resource to satisfy its new requirements or preferences [16].

Grid*W*ay also provides the application with fault tolerance capabilities by capturing GRAM callbacks, by periodically probing the GRAM job manager, and by inspecting the output of each job.

Grid*W*ay only relies on Globus services, so it could be used in any Grid infrastructure based on the Globus Toolkit, like IRISGrid. In the case of EGEE, Globus behavior has been slightly modified, but it does not lose its main protocols and interfaces, so it can be used in a standard way. In the following list, we detail the changes in the user-level middleware to adapt it to the EGEE and IRISGrid local configurations:

- *Resource broker*: the information schemas used by the EGEE and IRISGrid are the GLUE [13] and the Globus MDS schema, respectively. In this way, the Grid*W*ay resource selector was modified to gather information from the information services of the two Grids. The access to the GIIS (Grid Information Index Service) and BDII (Berkeley Database Information Service) servers is homogeneous, since both of them use an LDAP interface. The main difference between BDII and GIIS is that the information in BDII is stored in a persistent database, which is periodically updated. We have experienced that BDII can be much more reliable and efficient than GIIS.

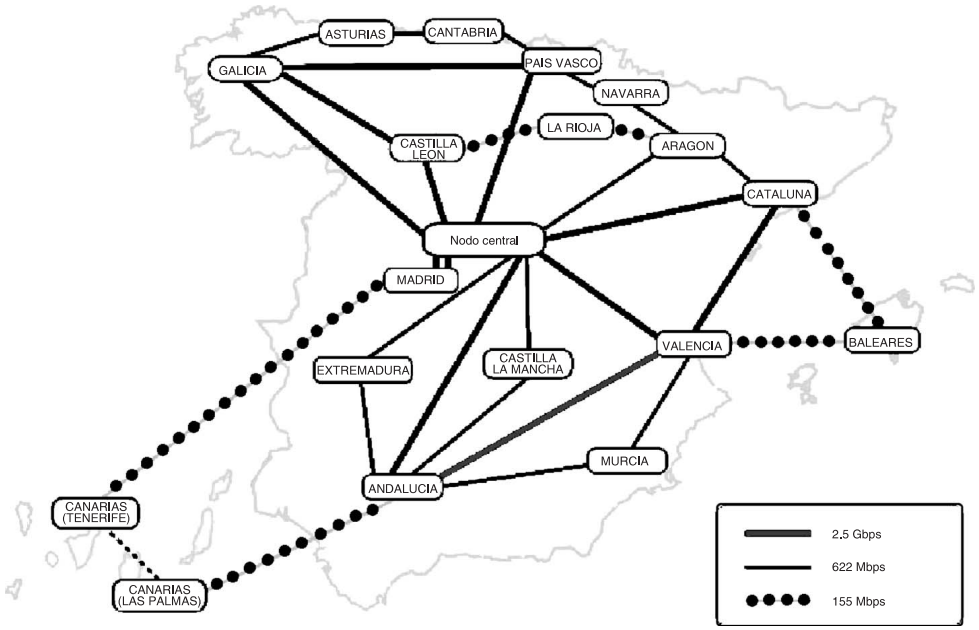


Fig. 2. Topology and bandwidths of RedIRIS-2.

Table 3
Core Grid middleware

Globus component	IRISGrid	EGEE
Security infrastructure	IRISGrid CA and manually generated `grid-mapfile`	DATAGRID-ES CA and automatically generated `grid-mapfile`
Resource management	GRAM with shared home directory in clusters	GRAM without shared home directory in clusters
Information services	IRISGrid GIIS and local GRIS, using the MDS schema	CERN BDII and local GRIS, using the GLUE schema
Data management	GASS and GridFTP	GASS and GridFTP

- *Job execution*: in the EGEE testbed the file transfers (i.e. executable, and input/output files) are initiated by a job wrapper running in the compute nodes, since shared file-systems are not supported for clusters. The Grid*Way* execution system was adapted to this special cluster configuration, by performing an explicit file staging between the front-end and the compute nodes in EGEE clusters.

5. Grid application: protein structure prediction

Bioinformatics, which has to do with the management and analysis of huge amounts of biological data, could enormously benefit from the suitability of the Grid to execute high-throughput applications. It is foreseeable that the Grid will be inevitably adopted, given that biological data are growing very fast, due to the proliferation of automated high-throughput experimental techniques and organizations dedicated to Biotechnology. Therefore, the resources required to manage and analyze these data will be soon available only from the Grid [15].

In this work, we consider a Bioinformatics application aimed at predicting the structure and thermodynamic properties of a target protein from its amino acid sequence. The algorithm, tested in the fifth round of Critical Assessment of techniques for protein Structure Prediction (CASP5), [6] aligns with gaps in the target sequence with all the 6150 non-redundant structures in the Protein Data Bank (PDB), [7] and evaluates the match between sequence and structure based on a simplified free energy function plus a gap penalty item. The lowest scoring alignment found is regarded as the prediction if it satisfies some quality requirements. In such cases, the algorithm can be used

[6] http://PredictionCenter.llnl.gov/casp5/
[7] http://www.pdb.org

Table 4
Experiments performed

Date (dd/mm/yy)	Start time (h:mm:ss)	End time (h:mm:ss)	Turnaround time (min)	Throughput (jobs/min)
22/10/2004	20:44:45	21:26:45	42.00	1.90
23/10/2004	23:58:32	0:32:22	33.82	2.37
24/10/2004	1:38:32	2:24:35	46.05	1.74
26/10/2004	13:44:03	14:34:25	50.37	1.59
26/10/2004	19:07:16	19:51:50	44.57	1.80

Table 5
Execution and transfer times (in seconds) per job on each IRISGrid resource

Host	Execution time		Transfer time	
	Mean	Dev.	Mean	Dev.
heraclito	2146	57	151	107
platon	919	275	67	50
descartes	611	33	48	30
socrates	647	103	51	27
aquila	1895	235	143	93
cepheus	2022	112	64	24
cygnus	755	74	33	20
babieca	1798	28	131	131
bw	697	176	123	54
llucalcari	1567	168	169	92
augusto	1200	233	89	61
caligula	1242	233	153	109
claudio	1228	184	187	131
lxsrv1	687	190	152	92

to estimate thermodynamic parameters of the target sequence, such as the folding free energy and the normalized energy gap [4].

To speed up the analysis and reduce the data needed, the PDB files are preprocessed to extract the contact matrices, which provide a reduced representation of protein structures. The algorithm is then applied twice, the first time as a fast search, in order to select the 200 best candidate structures, and the second time with parameters allowing a more accurate search of the optimal alignment.

We have applied the algorithm to the prediction of thermodynamic properties of families of orthologous proteins, i.e. proteins performing the same function in different organisms. If a representative structure of this set is known, the algorithm predicts it as the correct structure. The biological results of the comparative study of several families of orthologous proteins are presented elsewhere [28].

6. Experiences and results

The experiments presented in this section analyze a family of 80 orthologous proteins of the *Triose Phosphate Isomerase* enzyme. The experiment files consist of the executable (0.5 MB) provided for all the resource architectures in the testbed, the PDB files stored at the client and compressed (12.2 MB) to reduce the transfer time, the parameter files (1 KB), and the file containing the protein sequence to be analyzed (1 KB). Five experiments were conducted on different days during the same week, as shown in Table 4. The average turnaround time for the five experiments was 43.37 min.

Tables 5 and 6 reflect the transfer and execution times employed by each resource in analyzing one sequence of the problem. These metrics are useful to evaluate the impact of data movement strategies (e.g. file re-usage or replica selection and dissemination), individual resource performance or the influence of the interconnection network.

Given the dynamic nature of Grid environments it is important to quantify the fluctuations on these measurements. Thus, the standard deviation of the average of the transfer and execution times can be used as an indicator of the dynamism (if calculated on the same resource over time) and heterogeneity (if calculated at the same time over different resources) of the environment. In Tables 5 and 6, the standard deviation is an indicator of the dynamism, since it is calculated in an aggregated way for all the experiments.

As can be seen from Tables 5 and 6, resources with fast processors present a lower mean execution time ($\mu_{\text{lxsrv1}} = 687$, $\mu_{\text{bw}} = 697$, $\mu_{\text{grid003}} = 739$ and $\mu_{\text{lcgce02}} = 777$). Some resources also present a higher deviation in the execution time due to the queue system overhead and a slow front-end node ($\sigma_{\text{lcg02}} = 342$, $\sigma_{\text{ce00}} = 267$ and $\sigma_{\text{lcg2ce}} = 226$). The use of SMP nodes also causes a higher variability in the execution time when all the CPUs are simultaneously used ($\sigma_{\text{platon}} = 275$, $\sigma_{\text{augusto}} = 233$, $\sigma_{\text{caligula}} = 233$ and $\sigma_{\text{claudio}} = 184$) due to the contended use of shared resources, like memory and system buses. In the case of DIF-UM, all its three resources present

Table 6
Execution and transfer times (in seconds) per job on each EGEE resource

Host	Execution time		Transfer time	
	Mean	Dev.	Mean	Dev.
ce00	929	267	220	80
mallarme	945	191	123	68
lcg02	932	342	158	115
grid003	739	63	90	67
gtbcg12	1002	52	261	105
lcg2ce	889	226	208	113
lcgce02	777	179	98	68

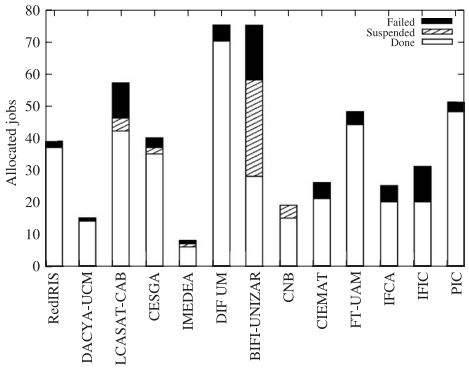


Fig. 3. Aggregated scheduling performed by Grid*Way* during the five experiments.

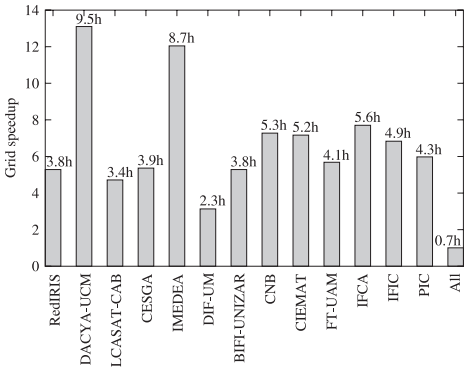


Fig. 4. Grid speedup (S_{Site}) and estimated turnaround time (T_{Site}) for each site.

two physical CPUs, but they report four logical CPUs due to hyper-threading. Therefore, four jobs were simultaneously scheduled to these resources, resulting in a great performance loss ($\mu_{\text{augusto}} = 1200$, $\mu_{\text{caligula}} = 1242$, $\mu_{\text{claudio}} = 1228$).

During the execution of each experiment, some of the jobs failed (core middleware failures) or were suspended for a long time in the local job manager (i.e SGE and PBS). Grid*Way* migrated these failed or suspended jobs to other resources. Fig. 3 shows the distribution of total executions and failures and job migrations over sites.

It is of crucial interest to analyze the potential gain in performance that a site could obtain by joining the Grid. To this end, let us define Grid speedup as

$$S_{\text{Site}} = \frac{T_{\text{Site}}}{T_{\text{Grid}}}, \tag{1}$$

where T_{Grid} is the turnaround time using all the Grid, and T_{Site} is the turnaround time using only the resources available in a given site. In this case we will estimate the optimum execution, time (T_{Site}), by calculating the optimum application *makespan* [17,22].

Fig. 4 shows each site speedup (the bars) and turnaround time (the values on top of bars). Notice that, since the number of jobs simultaneously submitted to the same resource have been limited to four, these metrics do not show the actual potential performance gain that could be obtained when using all the resources available on each organization. As shown in Fig. 4 those sites with modest computational resources present a higher Grid speedup.

Fig. 5. Testbed dynamic throughput during the five experiments and theoretical throughput of the most powerful site (DIF-UM).

Fig. 5 shows the dynamic throughput achieved during the five experiments alongside the theoretical throughput of the most

powerful site, where the problem could be solved in the lowest time, in this case DIF-UM (taking into account the above limitation in the number of simultaneously running jobs in a resource). The throughput achieved on each experiment varies considerably. This is due to the dynamic availability and load of the testbed. For example, resource ce00 at LCASAT-CAB was not available during the execution of the first experiment. Moreover, fluctuations in the load of network links and computational resources induced by non-Grid users affected to a lesser extent in the second experiment, as it was performed at midnight.

7. Conclusions

Loosely coupled Grids allow a straightforward resource sharing since resources are accessed and exploited through de facto standard protocols and interfaces, similar to the early stages of the Internet. This way, the *loosely coupled* model allows an easier, scalable and compatible deployment.

We have shown that the "end-to-end" principle works at the client side (i.e. the user-level Grid middleware) of a Grid infrastructure. Our proposed user-level Grid middleware, Grid*W*ay, is able to work with a standard core Grid middleware over any Grid fabric in a *loosely coupled* way. Moreover, since similar experiments have been previously performed at the resource side (i.e. the Grid fabric), we can conclude that the "end-to-end" principle could work on the two sides.

The straightforward process of integration of these largely different testbeds, although both are based on Globus, demonstrates that the Grid*W*ay approach (i.e. the Grid way), based on a modular, decentralized and "end-to-end" architecture, is appropriate for the Grid. Moreover, the evaluation of the resulting infrastructure shows that reasonable levels of performance and reliability are achieved.

Acknowledgments

We would like to thank all the institutions involved in the IRISGrid initiative and the EGEE project, in particular those who collaborated in the experiments. They are Red Académica y de Investigación Nacional (RedIRIS), Departamento de Arquitectura de Computadores y Automática (DACyA) at Universidad Complutense de Madrid (UCM), Laboratorio de Computación Avanzada, Simulación y Aplicaciones Telemáticas (LCASAT) at Centro de Astrobiología (CAB), Centro de Supercomputación de Galicia (CESGA), Instituto Mediterráneo de Estudios Avanzados (IMEDEA), Facultad de Informática (DIF) at Universidad de Murcia (UM), Instituto de Biocomputación y Física de Sistemas Complejos (BIFI) at Universidad de Zaragoza (UNIZAR), Instituto de Física de Cantabria (IFCA), Instituto de Física Corpuscular (IFIC), Centro Nacional de Biotecnología (CNB), Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT), Departamento de Física Teórica (FT) at Universidad Autónoma de Madrid (UAM) and Port d'Informació Científica (PIC).

We would like to also thank Ugo Bastolla, staff scientist in the Bioinformatics Unit at Centro de Astrobiología (CAB) and developer of the Bioinformatics application used in the experiments, for his support in understanding and modifying the application.

References

[1] R. Alfieri, et al., Virtual organization membership service, in: Proceedings of the 12th Hungarian Networkshop Conference, 2003.
[2] R.J. Allan, J. Gordon, A. McNab, S. Newhouse, M. Parker, Building overlapping grids, Technical Report, University of Cambridge, October 2003.
[3] M. Baker, R. Buyya, D. Laforenza, Grids and grid technologies for wide-area distributed computing, Software—Practice Experience 32 (15) (2002) 1437–1466.
[4] U. Bastolla, M. Vendruscolo, E.W. Knapp, A statistical mechanical method to optimize energy parameters for protein folding, Proc. Nat. Acad. Sci. USA 97 (2000) 3977–3981.
[7] E.B. Carpenter, RFC 1958: Architectural Principles of the Internet, category: Informational, June 1996.
[8] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, S. Tuecke, A Resource Management Architecture for Metacomputing Systems, Lecture Notes in Computer Science, vol. 1459, 1998, pp. 62–70.
[9] A. Delgado, P. Méndez, F. Donno, A. Sciabà, S. Campana, R. Santinelli, LCG-2 User Guide, 2004, available at ⟨http://edms.cern.ch/file/454439/1⟩.
[10] I. Foster, What is the Grid? A three point checklist, GRIDtoday 1(6), available at ⟨http://www.gridtoday.com/02/0722/100136.html⟩.
[11] I. Foster, C. Kesselman, Globus: a metacomputing infrastructure toolkit, Internat. J. Supercomput. Appl. 11 (2) (1997) 115–128.
[13] Glue Schema, 2002, available at ⟨http://www.hicb.org/glue/glue.htm⟩.
[15] E. Huedo, U. Bastolla, R.S. Montero, I.M. Llorente, Computational proteomics on the grid, New Generation Comput. 22 (2) (2004) 191–192.
[16] E. Huedo, R.S. Montero, I.M. Llorente, A framework for adaptive execution on grids, Internat. J. Software—Practice Experience (SPE) 34 (7) (2004) 631–651.
[17] E. Huedo, R.S. Montero, I.M. Llorente, Experiences on adaptive grid scheduling of parameter sweep applications, in: Proceedings of the 12th Euromicro Conference on Parallel Distributed and Network-based Processing (PDP2004), IEEE CS Press, 2004, pp. 28–38.
[18] O.S. José, L.M. Suárez, E. Huedo, R.S. Montero, I.M. Llorente, Resource performance management on computational grids, in: Proceedings of the Second International Symposium on Parallel and Distributed Computing (ISPDC 2003), 2003, pp. 215–221.
[19] A. Konstantinov, The NorduGrid grid manager and GridFTP server, Description and Administrator's Manual, available at ⟨http://www.nordugrid.org⟩.
[21] A. Merzky, et al., Simple API for grid applications—Strawman API, Global Grid Forum Report, 2005, available at ⟨http://wiki.cct.lsu.edu/saga/space/start⟩.
[22] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, second ed., Prentice-Hall, New Jersey, 2002.
[23] H. Rajic, et al., Distributed resource management application API specification 1.0, DRMAA Working Group—Global Grid Forum Report, 2003, available at ⟨http://www.drmaa.org⟩.
[24] J.M. Schopf, Ten actions when superscheduling, Technical Report GFD-I.4, Scheduling Working Group—The Global Grid Forum, 2001.
[25] J.M. Schopf, B. Nitzberg, Grids: the top ten questions, Sci. Programming, Grid Comput. 10 (2) (2002) 103–111 (Special Issue).
[26] O. Smirnova, P. Eerola, T. Ekelöf, et al., The NorduGrid architecture and middleware for scientific applications, in: Proceedings of the 11th International Conference on Computational Science, Lecture Notes in Computer Science, vol. 2657, 2003, pp. 264–273.
[28] R. van Ham, J. Kamerbeek, C. Palacios, C. Rausell, F. Abascal, U. Bastolla, J.M. Fernandez, L. Jimenez, M. Postigo, F. Silva, J. Tamames, E. Viguera, A. Latorre, A. Valencia, F. Morán, A. Moya, Reductive genome evolution in Buchnera Aphidicola, Proc. Nat. Acad. Sci. USA 100 (2003) 581–586.

José Luis Vázquez-Poletti received his M.E. in Computer Science (2004) from the Universidad Pontificia de Comillas (UPCo). He is doing his Ph.D. in Computer Architecture at the Universidad Complutense de Madrid (UCM). His research interests lie mainly in Grid Technology, in particular, metascheduling algorithms and their implementation.

Eduardo Huedo received his M.E. in Computer Science (1999) and Ph.D. in Computer Architecture (2004) from the Universidad Complutense de Madrid (UCM). He is Postdoctoral Researcher in the Advanced Computing Laboratory at Centro de Astrobiologia (CSIC-INTA), associated to NASA Astrobiology Institute. His research interests include Performance Management and Tuning, Parallel and Distributed Computing and Grid Technology.

Ruben S. Montero received his B.S. in Physics (1996), M.S in Computer Science (1998) and Ph.D. in Computer Architecture (2002) from the Universidad Complutense de Madrid (UCM). He is an Assistant Professor of Computer Architecture and Technology at UCM since 1999. He has held several research appointments at ICASE (NASA Langley Research Center), where he worked on computational fluid dynamics, parallel multigrid algorithms and Cluster computing. Nowadays, his research interests lie mainly in Grid Technology, in particular in adaptive scheduling, adaptive execution and distributed algorithms.

Ignacio M. Llorente received his B.S. in Physics (1990), M.S in Computer Science (1992) and Ph.D. in Computer Architecture (1995) from the Universidad Complutense de Madrid (UCM). He is Executive M.B.A. by Instituto de Empresa since 2003. He is an Associate Professor of Computer Architecture and Technology in the Department of Computer Architecture and System Engineering at UCM and Senior Scientist at Centro de Astrobiologia (CSIC-INTA), associated to NASA Astrobiology Institute. He has held several appointments since 1997 as a Consultant in High Performance Computing and Applied Mathematics at ICASE (NASA Langley Research Center). His research areas are Information Security, High-Performance Computing and Grid Technology.

A.2. A Comparison between Two Grid Scheduling Philosophies: EGEE WMS and GridWay

Cita Completa

J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *A Comparison Between two Grid Scheduling Philosophies: EGEE WMS and GridWay*. Multiagent and Grid Systems, vol. 3, n. 4, 2007, pp. 429-439.

Abstract

In order to achieve a reasonable degree of performance and reliability, Metascheduling has been revealed as a key functionality of the grid middleware. The aim of this paper is to provide a comparative analysis between two major grid scheduling philosophies: a semi-centralized approach, represented by the EGEE Workload Management System, and a fully distributed approach, represented by the GridWay Metascheduler. The distributed approach follow a loosely-coupled philosophy for the Grid resembling the *end-to-end* principle, which has fostered the spectacular development and diffusion of the Internet and, in particular, Web technologies in the past decade. The comparative is both theoretical, through a functionality checklist, and experimental, through the execution of a fusion physics plasma application on the EGEE infrastructure. This paper not only includes a standard analysis with the obtained times, but also a complex analysis based on a performance model.

Referencia de Citas Bibliográficas

Yu y Buyya (2005); Schopf (2001); Huedo et al. (2004); Baker et al. (2002); Foster y Kesselman (1997); Carpenter (1996); Rajic et al. (2003); Merzky et al. (2005); Smith (2003); Frey et al. (2002); Coleman et al. (2003); Thain et al. (2003); Wäldrich et al. (2005); Erwin y Snelling (2001); Vernugopal et al. (2006); Assunao et al. (2005); Luther et al. (2005); Campana et al. (2004); Huedo et al. (2006); Avellino et al. (2004); Morajko (2005); Huedo et al. (2006); Herrera et al. (2004); Herrera et al. (2007); Alfieri et al. (2005); Llorente et al. (2006); Castejón et al. (2004); Huedo et al. (2005); Vázquez-Poletti et al. (2006); Montero et al. (2006)

Notas Adicionales

La carencia de artículos que referencien a esta publicación y su propia inclusión en el JCR se debe a que esta revista es de reciente creación (2005).

Multiagent and Grid Systems – An International Journal 3 (2007) 429–439
IOS Press

A comparison between two grid scheduling philosophies: EGEE WMS and Grid *W*ay[1]

J.L. Vázquez-Poletti*, E. Huedo, R.S. Montero and I.M. Llorente
Departamento de Arquitectura de Computadores y Automática, Facultad de Informática, Universidad Complutense de Madrid, 28040 Madrid, Spain

Received 25 October 2006
Revised 20 February 2007
Accepted 24 May 2007

Abstract. In order to achieve a reasonable degree of performance and reliability, Metascheduling has been revealed as a key functionality of the grid middleware. The aim of this paper is to provide a comparative analysis between two major grid scheduling philosophies: a semi-centralized approach, represented by the EGEE Workload Management System, and a fully distributed approach, represented by the Grid *W*ay Metascheduler. The distributed approach follow a loosely-coupled philosophy for the Grid resembling the *end-to-end* principle, which has fostered the spectacular development and diffusion of the Internet and, in particular, Web technologies in the past decade. The comparative is both theoretical, through a functionality checklist, and experimental, through the execution of a fusion physics plasma application on the EGEE infrastructure. This paper not only includes a standard analysis with the obtained times, but also a complex analysis based on a performance model.

Keywords: Grid computing, metascheduling, EGEE WMS, Grid *W*ay, fusion physics

1. Introduction

The growing computational needs of nowadays projects have permitted the evolution to a new paradigm called Grid Computing. The ability to have applications draw computing power from a global resource pool to achieve high performance has become a new challenge for distributed-computing and Internet technologies. Several research centers share their computing assets in grids, which dramatically increase the number of processing and storage resources applications can access.

Different grid infrastructures are being deployed in the context of growing national and international research projects. Among the intervening elements of a computional grid, the Metascheduler is gathering most attention as a way to meet the challenging needs of several application domains. The term Metascheduler can be defined as a grid middleware that discovers, evaluates and allocates resources for grid jobs by coordinating activities between multiple heterogeneous schedulers that operate at local or cluster level [1]. In general, the scheduling process includes the following phases: resource discovery and selection; and job preparation, submission, monitoring, migration and termination [2].

Although we can find several philosophies for the Grid, discussed in Section 2, as well as different implementations of the Metascheduler, some of them enumerated in Section 3, the objective of this paper is to compare Grid *W*ay [3] and the EGEE Workload Management System (WMS) as two representative implementations. A short overview of their architectures can be found in Section 4 along with a comparative analysis of the functionality provided by both solutions. Then, Section 5 evaluates the fine-grain metrics and perfor-

*Corresponding author. Tel.: +34 91 394 75 41; Fax: +34 91 394 46 87; E-mail: jlvazquez@fdi.ucm.es.

[1]This research was supported by Consejería de Educación de la Comunidad de Madrid, Fondo Europeo de Desarrollo Regional (FEDER) and Fondo Social Europeo (FSE), through BIOGRIDNET Research Program S-0505/TIC/000101, and by Ministerio de Educación y Ciencia, through the research grant TIN2006-2806.

ISSN 1574-1702/07/$17.00

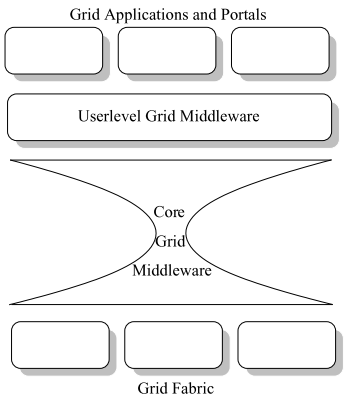


Fig. 1. *Loosely-coupled* Grid layers.

mance obtained by the two alternatives in the execution of a fusion physics plasma application on the EGEE infrastructure. Finally, the paper ends up with some conclusions and future work in Section 6.

2. A loosely-coupled philosophy for the grid

A Grid infrastructure is usually decomposed into the following layers [4]: Grid applications and portals; user-level Grid middleware; core Grid middleware; and Grid fabric. The two internal layers are called *the middleware*, since they connect applications with resources, or Grid fabric. In a *loosely-coupled* Grid, it is important to keep these layers separated and independent with a limited and well defined set of interfaces and protocols between them (Fig. 1).

The main characteristics of these environments are [4] autonomy (of the multiple administration domains), heterogeneity, scalability and dynamism. These properties completely determine the way that scheduling and execution on Grids have to be done. For example, scalability and autonomy prevent the deployment of centralized resource brokers with total control over client requests and resource status. On the other hand, the dynamic resource characteristics in terms of availability, capacity and cost make essential the ability to adapt job scheduling and execution to these conditions.

The Globus Toolkit [5] has become a de facto standard core middleware in Grid Computing. Globus services allow secure and transparent access to resources across multiple administrative domains, and serve as building blocks to implement the stages of Grid scheduling [2]. Globus architecture follows an hourglass model, which is indeed an *end-to-end* principle [6]. Therefore, instead of succumbing to the temptation of tailoring the core Grid middleware to its needs (since in such case the resulting infrastructure would be a highly distributed cluster), the *loosely-coupled* philosophy follows the *end-to-end* principle. Clients should have access to a wide range of resources provided through a limited and standardized set of protocols and interfaces. In the Grid, these are provided by the core Grid middleware, i.e. Globus. Just as, in the Internet, they are provided through the IP protocol.

The application of the *end-to-end* principle to Grid computing requires user-level middleware, such as Grid Way, in the client side to make it easier and more efficient the execution of applications. Such client middleware should provide the end user with portable programming paradigms and common interfaces, which are being standardized by the Open Grid Forum (OGF), see, for example the Distributed Resource Management Application API (DRMAA) [7] and the Simplified API for Grid Application (SAGA) [8].

The EGEE project is creating the largest production-level Grid Infrastructure in the world, which provides a level of performance and reliability never achieved before. A very restrictive set of requirements has been established for organizations that wish to take part in it. EGEE defines the user-level Grid middleware, the core Grid middleware and the Grid fabric, and so these layers are tightly related. EGEE uses the LCG (LHC Computing Grid)[2] middleware, LCG-2.

This architecture presents some limitations in terms of heterogeneity, as it has fixed configuration for clusters. The scalability of its deployment is also limited, as the middleware should be installed on the compute nodes, and they should have network connectivity. Also, LCG's focus is mainly on particle physics applications and its development is highly dependant on a single organization, CERN (the European Organization for Nuclear Research). Nevertheless, it is expected that the new EGEE middleware, gLite[3], will overcome to some extent some of these limitations.

3. Related work

Among the several implementations of the Metascheduler, the authors found representative CSF, Condor-

[2] http://lcg.web.cern.ch/.
[3] http://www.glite.org/.

G, VIOLA MetaScheduling Service and Gridbus Service Broker. CSF (Community Scheduler Framework) [9] is a WSRF-compliant Metascheduler built upon the Globus Toolkit. Depending if the access to a specific GRAM adapter is desired, the installation of LSF[4] is needed. It supports advance reservation booking and offers round-robin and reservation based scheduling algorithms.

Condor-G [10] is not conceived for supporting scheduling policies, but on the other hand, it supplies mechanisms that may be useful for a Metascheduler standing above, like *ClassAd* and *DAGMan*. *ClassAd* [11] (Classified Advertisement Language) allows the user to decide the resources for the job in the matchmaking process. *DAGMan* [12] (Directed Acyclic Graph Manager) is a workflow manager where interdependencies between jobs or data can be specified.

VIOLA [13] (Vertically Integrated Optical Testbed for Large Applications) is a project which aims to take benefit of an optical testbed using advanced techniques and services[5]. Its MetaScheduling Service stands on UNICORE [14] and provides support for complex distributed applications, as well as automation of coallocation of computational and network resources. Web-services allow to either implement adapters for local resource managers, and access the MetaScheduling Service itself.

Gridbus Service Broker [15] accesses transparently to various low level middleware solutions like Globus Toolkit, XGrid [16], UNICORE and Alchemi [17]. Its scheduling strategies consider both job deadline and budget specified by the user, in order to optimize the resource usage. On the other hand, custom scheduling algorithms can be implemented by means of writing a custom scheduler.

4. Grid scheduling infrastructures

The EGEE WMS and Grid Way are representative metascheduling technologies of different strategies to deploy a grid scheduling infrastructure. Grid Way follows the *loosely-coupled* Grid principle described in Section 2, mainly characterized by: autonomy of the multiple administration domains, heterogeneity, scalability and dynamism. A Grid Way instance is installed in each organization involved in the partner grid to provide scheduling functionality for intra-organization users (site-level scheduling). On the other hand, EGEE WMS provides a higher centralized strategy as one or more scheduling instances can be installed in the grid infrastructure, each providing scheduling functionality for a number of VOs (VO-level scheduling).

The EGEE WMS is the result of previous projects (Datagrid, CrossGrid). The present version of the EGEE WMS is based on the LCG-2 middleware and it is migrating to a new one called gLite which inherits many of the former elements and functionalities. LCG-2's WMS architecture, which is represented in Fig. 2, is highly centralized and each functionality is provided by almost a different machine, as it is conceived as a network service. The EGEE WMS components are the following: The User Interface (UI), which is from where the user sends the jobs; the Resource Broker (RB), which is based in Condor-G and uses most of its functionality; the Computing Element (CE) and the Worker Nodes (WN), which are the cluster frontend and the cluster nodes respectively, as it is established in the fixed configuration dictated by LCG-2's architecture; the Storage Element (SE), which is used for job files storage; the Logging and Bookkeeping service (LB), which registers job events [18].

After the user initiates his use of the Grid by authenticating from the UI and the job parameters are defined, the RB receives the input files through an *input sandbox*. The RB contacts the Information Service, where resource (CE and SE) information is published. If needed by the job, the Catalog Service is also queried by the RB. Then, the Job Submission Service receives an expanded version of the job description, result of the scheduling decisions taken by the RB (matchmaking process), so it submits the job to the chosen CE. In the meanwhile, the RB submits to that CE the *input sandbox* and its contact information. The job then is sent to a WN where it is executed. When the job is done, the *ouput sandbox* is sent back to the RB and then, to the user when requested. In any case, there are several alternative brokering services to submit the job.

Grid Way works on top of Globus services, performing job execution management and resource brokering, allowing unattended, reliable, and efficient execution of jobs, array jobs, or complex jobs on heterogeneous, dynamic and *loosely-coupled* Grids formed by Globus resources. Grid Way's modular architecture, represented in Fig. 2, is conformed by the Grid Way Daemon (GWD) and different Middleware Access Drivers (MADs) to access different Grid Services (resource information, job execution and file transfer), all of them

[4] http://www.platform.com/Products/Platform.LSF.Family/Platform.LSF/.
[5] http://www.viola-testbed.de/.

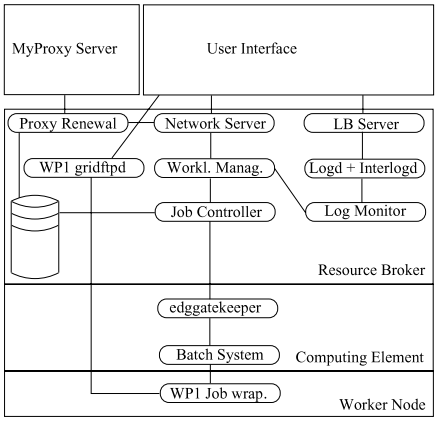


Fig. 2. Architecture of LCG-2's WMS.

in just one host, as Grid Way is conceived as a client tool. Grid Way performs all the job scheduling and submission steps transparently to the end user adopting job execution to changing Grid conditions by providing fault recovery mechanisms, dynamic scheduling, migration on-request and opportunistic migration. A typical job cycle in Grid Way is as explained ahead. After the user provides the GWD with the job description, the resource selection is performed. Once a decision is made, the needed execution and transfer MADs are loaded. The first stage is the *prolog*, where the remote system is prepared by creating a job directory and staging the input files. Then comes the *wrapper* where the real job execution takes place. Finally, during the *epilog* stage, the output files are staged back and the remote directory is cleared. Grid Way allows the deployment of organization-level metaschedulers that provide support for multiple intra-organization users in each scheduling instance. There is one scheduling instance for each organization and all instances compete with each other for the available resources.

4.1. Scheduling capabilities

Both LCG-2 WMS and Grid Way treat jobs in a FIFO way and support dynamic scheduling, providing a way to filter and evaluate resources based on dynamic attributes, by means of requirement and rank expressions. In LCG-2 the names of these attributes are the same as retrieved from the information service. Nevertheless, in Grid Way , these expressions are based on common resource attributes, independent from the information service, providing another way of decoupling. While LCG-2 RB accesses only the BDII servers and only processes the GLUE Schema, Grid Way's different information MADs allow to access the most common information services. Considering execution and transfer functionalities, and using the adoption of interfaces and protocols based on Web Services (WS) for this comparison, both LCG-2 and Grid Way support Globus pre-WS services, but only Grid Way allows access to Globus WS services [19].

Grid Way supports opportunistic migration. That is, each scheduling cycle evaluates the benefits of migrating submitted jobs to new available resources (recently added or freed) by comparing rank values. In LCG-2, this functionality is not supported and the ranking only affects submission.

Considering performance slowdown detection, Grid Way takes count of the suspension time in remote batch systems and requests a migration when it exceeds a given threshold. Moreover, jobs are submitted together with a light-weight self monitoring system. The job will migrate when it doesn't receive as much CPU as the user expected. None of the performance slowdown detection mechanisms given by Grid Way are implemented in LCG-2. The monitoring in the LCG-2 architecture is provided by the LB, which records only basic job states and mixes them with events originated in other components.

With Grid Way , an application can take decisions about resource selection as its execution evolves by modifying its requirement and rank expressions and requesting a migration. In LCG-2 RB instead, these expressions are only set at the beginning.

In LCG-2, the PBS Event Logging (APEL) is employed for distributed accounting [20]. Grid Way gives the user local accounting functionalities, standing on the Berkeley Database.

4.2. Fault detection and recovery capabilities

The LCG-2 RB incorporates error detection mechanisms provided by Condor-G [21]. On the other hand, Grid Way detects job cancellation (when the job exit code is not specified), remote system crash and network disconnection (both when the polling of the job fails). In all of these cases, Grid Way requests a migration for the job [22].

LCG-2 WMS supports checkpointing by providing an API to allow applications to be instrumented to save the state of the process (represented by a list of variable

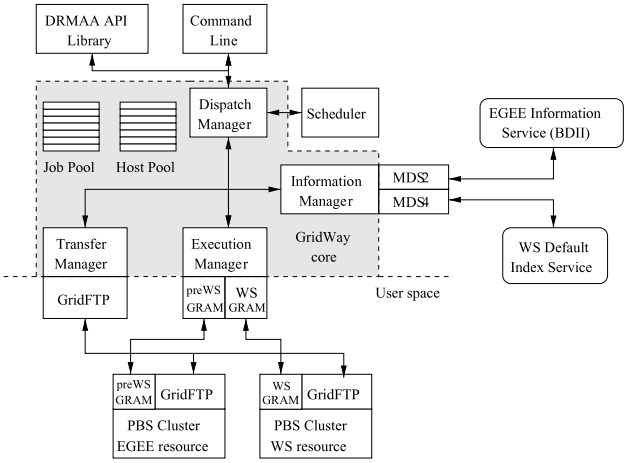


Fig. 3. Architecture of Grid *Way*.

and value pairs) at any moment during the execution of a job. Also, it provides the mechanisms to restart the computation from checkpointed data (previously saved state). If a job fails, the WMS automatically reschedules the job and resubmits it to another compatible resource. There, the last state is automatically retrieved and the computation is restarted. The user can also retrieve the saved state for a later manual resubmission, where the user can specify if the job must start from this retrieved checkpoint data. With Grid *Way*, user-level checkpointing or architecture independent restart files managed by the programmer must be implemented. Migration is implemented by restarting the job on the new candidate host. If the checkpointing files are not provided, the job should be restarted from the beginning. These checkpoints are periodically retrieved to the client machine or a checkpoint server.

Also the system running the scheduler could fail. Grid *Way* persistently saves its state in order to recover or restart the jobs when the system is restarted. LCG-2 RB relies on Condor-G, which stores persistently all relevant state for each submitted job in the scheduler's job queue [10].

4.3. User interface functionality

Both Grid *Way* and LCG-2 RB allow single jobs. The LCG-2 RB can handle jobs with dependencies (DAGMan functionality) and interactive jobs. On the other hand, Grid *Way* allows array jobs and jobs with dependencies. For providing this last job type support, Grid *Way* gives the user a functionality to synchronize jobs. In the case of LCG-2, synchronization must be implemented by a periodical polling (active wait), and job dependencies is supported by the Condor's *DAGMan* tool [20]. A *DAGMan* process is locally spawned for each Direct Acyclic Graph (DAG) submitted to Condor-G.

Focusing on the command line interface, both Grid *Way* and LCG-2 RB give the user full control of his jobs. Anyway, Grid *Way* incorporates commands which allow the user to migrate and synchronize jobs, functionalities not provided by LCG-2.

Also, Grid *Way* offers C and Java implementations of the DRMAA Application Programming Interface, which is a Open Grid Forum (OGF) standard [23,24]. The EDG WMS API given by LCG-2[6] is not standard.

4.4. Installation and configuration issues

Both Grid *Way* and LCG-2 RB are modular. The LCG-2 RB is a network service and stands over a great list of external dependencies (requires more other services than Globus): Condor-G, MySQL, etc. Grid *Way* stands on the Globus middleware at the client

[6] http://www.to.infn.it/grid/workload_management/apiDoc/edg-wms-api-index.html.

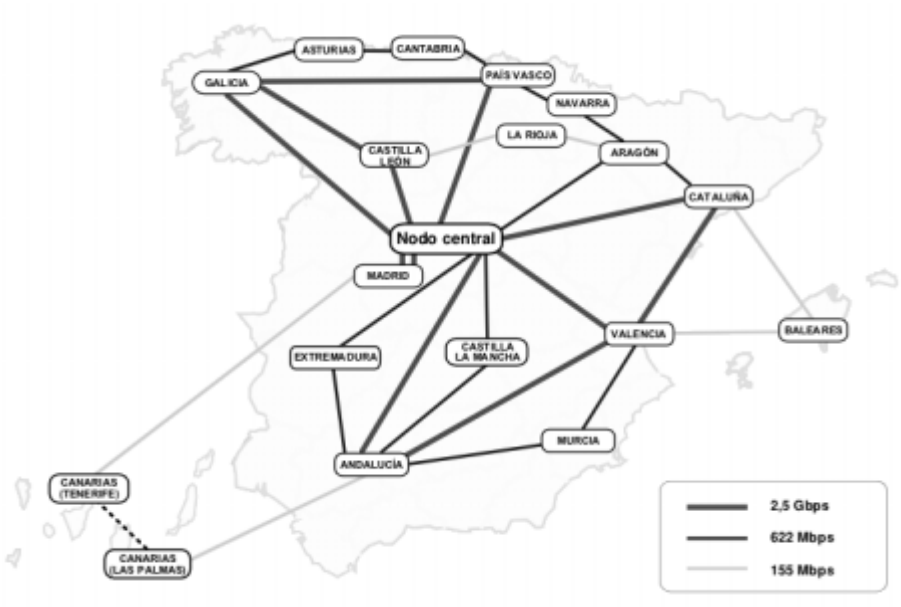


Fig. 4. Topology and bandwidths of RedIRIS-2.

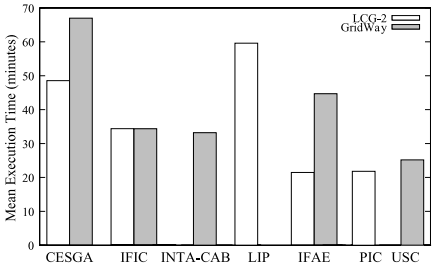


Fig. 5. Mean T_{exe} per Site.

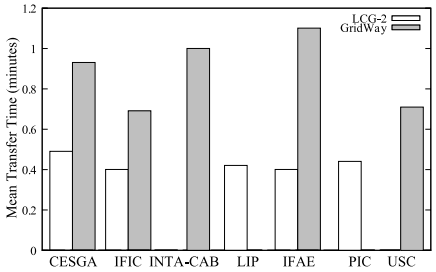


Fig. 6. Mean T_{xfr} per Site.

side and could be extended or used as a building block for more complex architectures that implement service-level agreements (SLAs) or advanced reservation. Also, LCG-2 is encouraged to be installed in Scientific Linux machines. Nevertheless, other flavors of Linux are being used.

Analyzing security mechanisms, LCG-2 takes them from the Globus Toolkit GSI and also implements the VOMS, where an user name and role are used (it manages the authorization information in the scope of the VO). VOMS allows a fine grained control of the use of the resources both to the users' organization and to the resource owners [25].

Grid Way can be installed to implement several Grid architectures, namely: enterprise Grids, to enable diverse resource sharing to improve internal collaboration and achieve a better return from IT investment; partner Grids (like the EGEE infrastructure described

Table 1
EGEE grid resources employed during the experiment

Site	Processor	Speed (GHz)	Nodes	DRMS
CESGA	Intel Pentium III	1.1	46	PBS
IFAE	Intel Pentium 4	2.8	11	PBS
IFIC	AMD Athlon	1.2	127	PBS
INTA-CAB	Intel Pentium 4	2.8	4	PBS
LIP	Intel Xeon	2.8	25	PBS
PIC	Intel Pentium 4	2.8	172	PBS
USC	Intel Pentium III	1.1	100	PBS

here) to provide large-scale, secure and reliable sharing of resources among partner organizations; and utility Grids [26] to provide access to the latest computing platform and technology and still be flexible enough to adjust capacity as required without needing to purchase costly hardware.

5. Experimental conditions and results

The Grid infrastructure used for the experiments is the corresponding to the test Virtual Organization at the Southwest Federation (SWETEST VO) of the EGEE project (Table 1). All Spanish sites are connected by RedIRIS, the Spanish Research and Academic Network, whose interconnection links of the different nodes are shown in Fig. 4.

The target application, called *Truba*, performs the tracing of a single ray of a microwave beam launched inside a fusion reactor [27]. Each experiment involves the execution of 50 instances of the *Truba* application. The experiments were performed with a development version of *Truba*, whose average execution time on a Pentium 4 of 3.20 GHz is 9 minutes. *Truba*'s executable file size is 1.8 MB, input file size is 70 KB, and output file size is about 549 KB.

For the EGEE WMS experiments, it has been developed a framework using the lcg2.1.69 User Interface C++ API, which provides support to submit, monitor and control each single ray tracing application to the grid. This framework works in the following way: First of all, a launcher script generates the JDL files needed. Then, the framework launches all the single ray tracing jobs simultaneously, periodically querying each job's state. And finally, it retrieves the job's output. The scheduling decisions are of course delegated to the EGEE WMS.

Grid Way only relies on Globus services, so it could be used in any Grid infrastructure based on the Globus Toolkit, both pre-WS and WS [28]. In the case of EGEE (LCG-2), Globus behaviour has been slightly modified, but it does not loose its main protocols and interfaces, so Grid Way can be used in a standard way to access LCG-2 resources [29].

In both cases, the jobs were submitted from Universidad Complutense de Madrid. The RB employed for the experiments with LCG-2 was located at the IFIC site and used an eager scheduling policy.

5.1. Experimental results

Table 2 shows a summary of the performance exhibited by the two scheduling systems in the execution of the fusion application. Additionally, Fig. 5 shows the mean execution time per site and Fig. 6, the mean transfer time per site. As can be seen, Grid Way presents a higher transfer time, because additional jobs are submitted for the *prolog* and *epilog* stages [3]. However, the experiments were conducted with a Grid Way version previous to 4.7, in which this issue was solved. On the other hand, the standard deviation of raw performance metrics can be interpreted as an indicator of the heterogeneity in the grid resources and interconnection links [29]. Finally, the lower overhead induced by Grid Way shows the benefits of its lighter approach and the functionality for performance slowdown detection.

The EGEE WMS spent 195 minutes (3.25 hours) to execute the 50 jobs, giving a productivity equal to 15.38 jobs/hour. Grid Way spent 120 minutes (2 hours) to execute the same workload, giving a productivity equal to 25 jobs/hour. The conclusion is that Grid Way takes better advantage of the available resources due to its superior scheduling capabilities on dynamic resources. In fact, during the experiments with the EGEE WMS, several problems described before were evidenced. The LCG-2 RB does not provide support for opportunistic migration and slowdown detection, and jobs are assigned to busy resources.

Additionally, the achieved level of parallelism [30] can be obtained by using the following expression:

$$U = \frac{T_{exe}}{T}, \tag{1}$$

being T_{exe} the sum of job execution times and T the turnaround time. The level of parallelism achieved by Grid Way was higher than the level achieved by the EGEE WMS (14.91 and 6.89 respectively).

Not all jobs ended successfully at the first try. In the case of the EGEE WMS, 31 jobs were affected and they had to be resubmitted. However, with Grid Way , only 1 job failed, but there were 21 migrations mostly due to suspension timeouts (too much delay in a queue), and better resource discovery (too much time allocated to a resource when better resources are waiting to be used).

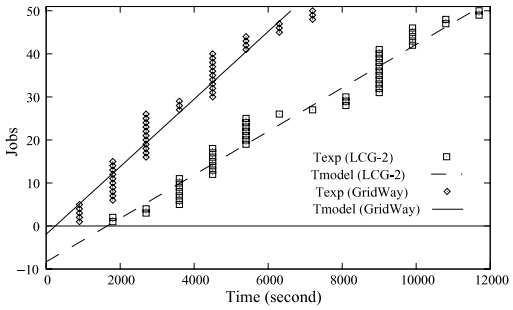


Fig. 7. Measurements of r_∞ and $n_{1/2}$ parameters for both platforms.

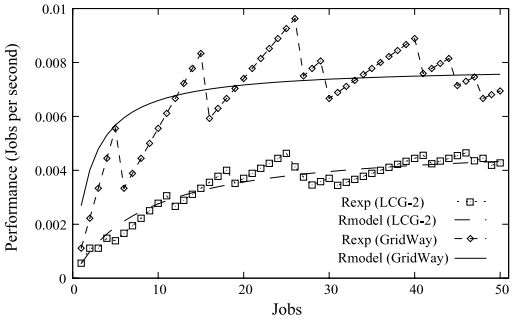


Fig. 8. Experimental and predicted performance.

A methodology to analyze the performance of computational Grids in the execution of high throughput computing applications has been proposed in [31]. This performance model enables the comparison of different platforms in terms of the following parameters: asymptotic performance (r_∞), which is the maximum rate of performance in tasks executed per second, and half-performance length ($n_{1/2}$), which is the number of tasks required to obtain half of the asymptotic performance. A first order characterization of a grid by means of these parameters is:

$$n(t) = r_\infty t - n_{1/2}. \quad (2)$$

Then, the performance of the system, jobs completed per second, can be defined with a finite number of tasks with:

$$r(n) = n(t)/t = \frac{r_\infty}{1 + n_{1/2}/n}, \quad (3)$$

where n is the number of jobs. The parameters of the model, r_∞ and $n_{1/2}$, are obtained by linear fitting to the experimental results obtained in the execution of the applications.

Figures 7 and 8 show the experimental performance obtained with the two workload management systems, along with that predicted by Eqs (2) and (3). With EGEE WMS, r_∞ was 0.0051 jobs/second (18.19 jobs/hour) and $n_{1/2}$ was 8.33. With Grid Way , r_∞ was 0.0079 jobs/second (28.26 jobs/hour) and $n_{1/2}$ was 1.92. From the different values of $n_{1/2}$, it can be deduced that Grid Way needs less jobs to obtain half of the asymptotic performance due to an earlier job allocation in the resources.

6. Conclusions and future work

Metascheduling has been proved to be a needed step in the evolution of grid middleware. In this paper, some implementations of the Grid Metascheduler concept

Table 2
Performance metrics for both platforms

	T_{exe} (m)		T_{xfr} (m)		T	Prod.	Ovh.
	Mean	Dev.	Mean	Dev.	(m)	(j/h)	(m/j)
LCG-2	30.33	11.38	0.42	0.06	195	15.38	1.82
Grid Way	36.80	16.23	0.87	0.51	120	25.00	0.52

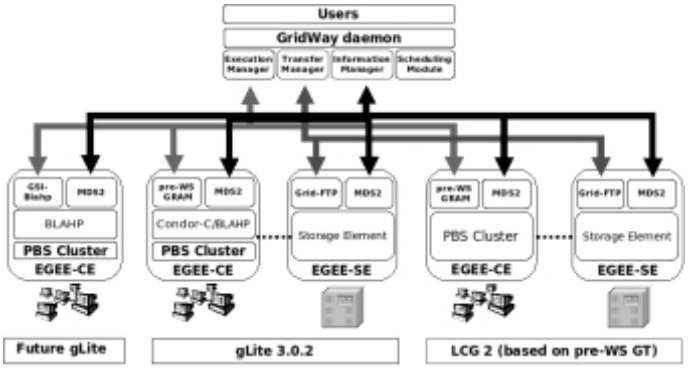


Fig. 9. Interaction of Grid Way with different EGEE middleware.

have been reviewed. Due to continuity in the research line, the authors have chosen to compare exhaustively the EGEE WMS and Grid Way. The target application to be ported onto the Grid in order to gain experimental results pertains to fusion physics, a new Grid technology demanding area.

There has been demonstrated that Grid Way offers a Metascheduling alternative for users, application developers and Grid managers, to the LCG-2 RB available in the EGEE distribution. Grid Way provides compatibility to applications with DRM systems that implement the DRMAA standard. The command line interface is similar to that found on local resource managers to submit, kill, migrate, monitor and synchronize single, array and interdependent jobs. Moreover, deployment and maintenance of Grid Way is not only easy and fast, also a wide variety of platforms is supported. Finally, Grid Way achieves lower overhead and higher productivity than the EGEE WMS, mostly because it reduces the number of job submission stages and provides mechanisms, not given by the LCG-2 RB, such as opportunistic migration and performance slowdown detection that considerably improve the usage of the resources. Nevertheless, LCG-2 provides other components that weren't considered in this article, such as data management.

As future work, Grid Way will continue its evolution, standing on its modular architecture. Its commitment is to support the access to the different middleware provided by EGEE (see Fig. 9).

Acknowledgments

This work makes use of results produced by the Enabling Grids for E-sciencE project, a project co-funded by the European Commission (under contract number INFSO-RI-031688) through the Sixth Framework Programme. EGEE brings together 91 partners in 32 countries to provide a seamless Grid infrastructure available to the European research community 24 hours a day. Full information is available at http://www.eu-egee.org/.

The authors would like to thank all the institutions involved in the EGEE project, in particular those who collaborated in the experiments.

References

[1] J. Yu and R. Buyya, A Taxonomy of Workflow Management Systems for Grid Computing, *Journal of Grid Computing* **3** (2005), 171–200.

[2] J.M. Schopf, Ten Actions when Superscheduling, Technical Report GFD.4, The Open Grid Forum (2001) Scheduling Working Group.
[3] E. Huedo, R.S. Montero and I.M. Llorente, A Framework for Adaptive Execution on Grids, *Software – Practice and Experience* **34** (2004), 631–651.
[4] M. Baker, R. Buyya and D. Laforenza, Grids and Grid Technologies for Wide-Area Distributed Computing, *Software – Practice and Experience* **32** (2002), 1437–1466.
[5] I. Foster and C. Kesselman, Globus: A Metacomputing Infrastructure Toolkit, *Intl J Supercomputer Applications* **11** (1997), 115–128.
[6] B. Carpenter, *E.: RFC 1958: Architectural Principles of the Internet* (1996), Category: Informational.
[7] H. Rajic, R. Brobst, W. Chan, J. Gardiner, A. Haas, B. Nitzberg and J. Tollefsrud, Distributed Resource Management Application API Specification 1.0. Document GFD.22, *The Open Grid Forum* (2003), DRMAA Working Group.
[8] A. Merzky, S. Jha, T. Goodale, J. Shalf and C. Smith, *Simple API for Grid Applications – Strawman API*, Document RC 4, The Open Grid Forum, 2005 Available at http://wiki.cct.lsu.edu/saga/space/start/.
[9] C. Smith, Open Source Metascheduling for Virtual Organizations with the Community Scheduler Framework Technical report, Platform Computing Inc. 2003.
[10] J. Frey, T. Tannenbaum, M. Livny, I. Foster and S. Tuecke, Condor-G: A Computation Management Agent for Multi-Institutional Grids, *Cluster Computing* **5** (2002), 237–246.
[11] N. Coleman, R. Raman, M. Livny and M. Solomon, Distributed Policy Management and Comprehension with Classified Advertisements, Document UW-CS-TR-1481, University of Wisconsin – Madison Computer Sciences Department, 2003.
[12] D. Thain, T. Tannenbaum and M. Livny, Condor and the Grid, in: *Grid Computing John Wiley & Sons Inc*, 2003, pp. 299–335.
[13] O. Wldrich, W. Ziegler and P. Wieder, A Meta-Scheduling Service for Co-allocating Arbitrary Types of Resources, Technical Report TR-0010, Institute on Resource Management and Scheduling, CoreGRID – Network of Excellence, 2005.
[14] D.W. Erwin and D.F. Snelling, UNICORE: A Grid Computing Environment. in: *Proc. 7th Intl. Conf. Parallel Processing* (*Euro-Par 2001*), (Vol. 2150) of Lecture Notes in Computer Science, 2001, pp. 825–834.
[15] S. Venugopal, R. Buyya and L. Winton, A Grid Service Broker for Scheduling e-Science Applications on Global Data Grids, *Concurrency and Computation: Practice and Experience* **18** (2006), 685–699.
[16] M. de Assunao, K. Nadiminti, S. Venugopal, T. Ma and R. Buyya, An Integration of Global and Enterprise Grid Computing: Gridbus Broker and Xgrid Perspective. in: *Proc. 4th International Conference on Grid and Cooperative Computing*, (Volume 3795) of Lecture Notes in Computer Science, 2005, pp. 406–417.
[17] A. Luther, R. Buyya, R. Ranjan and S. Venugopal, *Alchemi: A .NET-Based Enterprise Grid Computing System*, in: Proc. 6th International Conference on Internet Computing (ICOMP'05), 2005, 269–278.
[18] S. Campana, M. Litmaath and A. Sciaba, *LCG-2 Middleware Overview*, Available at https://edms.cern.ch/document/498079/0.1, 2004.
[19] E. Huedo, R.S. Montero and I.M. Llorente, A Modular Architecture for Interfacing Pre-WS and WS Grid Resource Management, *Future Generation Computing Systems* **23** (2006), 252–261.
[20] G. Avellino, S. Beco, B. Cantalupo et al., The DataGrid Workload Management System: Challenges and Results, *Journal of Grid Computing* **2** (2004), 353–367.
[21] A. Morajko, E. Fernandez, A. Fernandez, E. Heymann and M.A. Senar, Workflow Management in the CrossGrid Project, in: *Proc. European Grid Conference* (*EGC2005*), (Volume 3470) of Lecture Notes in Computer Science, 2005, 424–433.
[22] E. Huedo, R.S. Montero and I.M. Llorente, Evaluating the Reliability of Computational Grids from the End User's Point of View, *Journal of Systems Architecture* **52** (2006), 727–736.
[23] J. Herrera, E. Huedo, R.S. Montero and I.M. Llorente, Developing Grid-Aware Applications with DRMAA on Globus-based Grids, in: *Proc. 10th Intl. Conf. Parallel Processing* (*Euro-Par 2004*), (Volume 3149) of Lecture Notes in Computer Science, 2004, pp. 429–435.
[24] J. Herrera, E. Huedo, R.S. Montero and I.M. Llorente, GridWay DRMAA 1.0 Implementation – Experience Report. Document GFD.104, *The Open Grid Forum* (2007), DRMAA Working Group.
[25] R. Alfieri, R. Cecchini et al., From gridmap-file to VOMS: Managing Authorization in a Grid Environment, *Future Generation Computing Systems* **21** (2005), 549–563.
[26] I.M. Llorente, R.S. Montero, E. Huedo and K. Leal, A Grid Infrastructure for Utility Computing. in: Proc. 3rd International Workshop on Emerging Technologies for Next-generation GRID (ETNGRID 2006), 15th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2006), *IEEE Computer Society* (2006), 163–168.
[27] F. Castejon, M.A. Tereshcenko, et al., Electron Bernstein Wave Heating Calculations for TJ-II Plasmas, *American Nuclear Society* **46** (2004), 327–334.
[28] E. Huedo, R.S. Montero and I.M. Llorente, Coordinated Use of Globus Pre-WS and WS Resource Management Services with GridWay, in: it Proc. 2nd Workshop on Grid Computing and its Application to Data Analysis (GADA'05) On The Move Federated Conferences, (Volume 3762) of Lecture Notes in Computer Science, 2005, pp. 234–243.
[29] J.L. Vázquez-Poletti, E. Huedo, R.S. Montero and I.M. Llorente, Coordinated Harnessing of the IRISGrid and EGEE Testbeds with GridWay, *Journal of Parallel and Distributed Computing* **66** (2006), 763–771.
[30] E. Huedo, R.S. Montero and I.M. Llorente, An Evaluation Methodology for Computational Grids, in: *Proc. 2005 International Conference on High Performance Computing and Communications* (*HPCC05*), (Volume 3726) of Lecture Notes in Computer Science, 2005, pp. 499–504.
[31] R.S. Montero, E. Huedo and I.M. Llorente, Benchmarking of High Throughput Computing Applications on Grids, *Parallel Computing* **32** (2006), 267–279.

Authors' Bios

José Luis Vázquez-Poletti received his M.E. in Computer Science (2004) from the Universidad Pontificia de Comillas (UPCo). He is doing his Ph.D. in Computer Architecture at the Universidad Complutense de Madrid (UCM). He is a Teaching Assistant of Computer Architecture and Technology in the Department

of Computer Architecture and System Engineering at UCM since 2006. His research interests lie mainly in Grid Technology, in particular, metascheduling algorithms and their implementation.

Eduardo Huedo received his M.E. in Computer Science (1999) and Ph.D. in Computer Architecture (2004) from Universidad Complutense de Madrid (UCM). He is an Assistant Professor of Computer Architecture and Technology in the Department of Computer Architecture and System Engineering at UCM since 2006. Previously, he was Postdoctoral Researcher in the Advanced Computing Laboratory at Centro de Astrobiología (CSIC-INTA), associated to NASA Astrobiology Institute. His research interests include Performance Management and Tuning, Parallel and Distributed Computing and Grid Technology.

Rubén S. Montero received his B.S. in Physics (1996), M.S. in Computer Science (1998) and Ph.D. in Computer Architecture (2002) from the Universidad Complutense de Madrid (UCM). He is an Associate Professor of Computer Architecture and Technology in the Department of Computer Architecture and System Engineering at UCM since 2006. He has held several research appointments at ICASE (NASA Langley Research Center), here he worked on computational fluid dynamics, parallel multigrid algorithms and Cluster computing. Nowadays, his research interests lie mainly in Grid Technology, in particular in adaptive scheduling, adaptive execution and distributed algorithms.

Ignacio M. Llorente received his B.S. in Physics (1990), M.S. in Computer Science (1992) and Ph.D. in Computer Architecture (1995) from the Universidad Complutense de Madrid (UCM). He is Executive M.B.A. by Instituto de Empresa since 2003. He is Full Professor of Computer Architecture and Technology in the Department of Computer Architecture and System Engineering at UCM and Senior Scientist at Centro de Astrobiología (CSIC-INTA), associated to NASA Astrobiology Institute. He has held several appointments since 1997 as a Consultant in High Performance Computing and Applied Mathematics at ICASE (NASA Langley Research Center). His research areas are Information Security, High-Performance Computing and Grid Technology.

A.3. Execution of a Bioinformatics Application in a Joint IRISGrid/EGEE Testbed

Cita Completa

J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *Execution of a Bioinformatics Application in a Joint IRISGrid/EGEE Testbed*. LaSCoG Workshop, 6th PPAM Conference, Poznan (Polonia), Septiembre 2005. Lecture Notes in Computer Science (LNCS). Vol. 3911, pp. 831-838, 2006. Springer Verlag.

Abstract

This paper describes the execution of a Bioinformatics application over the largest ever spanish testbed. This testbed is composed of resources devoted to EGEE and IRISGrid projects and has been integrated by taking advance of the modular, decentralized and *end-to-end* architecture of the GridWay framework. Results show the feasibility of building loosely-coupled Grid environments only based on Globus services, while obtaining non trivial levels of quality of service. Such approach allows a straightforward resource sharing as the resources are accessed by using *de facto* standard protocols and interfaces.

Referencia de Citas Bibliográficas

Foster (2002); Baker et al. (2002); Allan et al. (2003); José et al. (2003); Schopf (2002); Carpenter (1996); Schopf (2001); Huedo et al. (2004); Berman et al. (2003); Buyya et al. (2002); Frey et al. (2002); Vadhiyar et al. (2003); Huedo et al. (2005); Bastolla et al. (2004)

Tasa de Aceptación

Este congreso recibió 130 contribuciones, de los cuales se aceptaron 50 (un 38 %).

Execution of a Bioinformatics Application in a Joint IRISGrid/EGEE Testbed*

José Luis Vázquez-Poletti[1], E. Huedo[2],
Rubén S. Montero[1], and Ignacio M. Llorente[1,2]

[1] Departamento de Arquitectura de Computadores y Automática. Facultad de Informática, Universidad Complutense de Madrid. 28040 Madrid, Spain
[2] Laboratorio de Computación Avanzada, Simulación y Aplicaciones Telemáticas, Centro de Astrobiología (CSIC-INTA), 28850 Torrejón de Ardoz, Spain

Abstract. This paper describes the execution of a Bioinformatics application over a highly distributed and heterogeneous testbed. This testbed is composed of resources devoted to EGEE and IRISGrid projects and has been integrated by taking advantage of the modular, decentralized and "end-to-end" architecture of the Grid*W*ay framework. Results show the feasibility of building loosely-coupled Grid environments based only on Globus services, while obtaining non trivial levels of quality of service. Such approach allows a straightforward resource sharing as the resources are accessed by using *de facto* standard protocols and interfaces.

1 Introduction

Different Grid infrastructures are being deployed within growing national and transnational research projects. The final goal of these projects is to provide the end user with much higher performance than that achievable on any single site. However, from our point of view, it is arguable that some of these projects embrace the Grid philosophy, and to what extent. This philosophy, proposed by Foster [1], defines *a Grid* as a system (i) not subject to a centralized control and (ii) based on standard, open and general-purpose interfaces and protocols, (iii) while providing some level of quality of service (QoS), in terms of security, throughput, response time or the coordinated use of different resource types. In current projects, there is a tendency to ignore the first two requirements in order to get higher levels of QoS. However, these requirements are even more important because they are the key to the success of *the Grid*.

The Grid philosophy leads to computational environments, which we call *loosely-coupled Grids*, mainly characterized by [2]: autonomy (of the multiple administration domains), heterogeneity, scalability and dynamism. In a loosely-coupled Grid, the different layers of the infrastructure should be separated from

* This research was supported by Ministerio de Ciencia y Tecnología, through the research grants TIC 2003-01321 and 2002-12422-E, and by Instituto Nacional de Técnica Aeroespacial "Esteban Terradas" (INTA) – Centro de Astrobiología. The authors participate in the EGEE project, funded by the European Union under contract IST-2003-508833.

R. Wyrzykowski et al. (Eds.): PPAM 2005, LNCS 3911, pp. 831–838, 2006.

each other, being only communicated with a limited and well defined set of interfaces and protocols. This layers are [2]: Grid fabric, core Grid middleware, user-level Grid middleware, and Grid applications.

The coexistence of several projects, each with its own middleware developments, adaptations or extensions, arise the idea of using them simultaneously (from an user's viewpoint) or contribute the same resources to more than one project (from an administrator's viewpoint). One approach could be the development of gateways between different middleware implementations [3]. Other approach, more in line with the Grid philosophy, is the development of client tools that can adapt to different middleware implementations. We hope this could lead to a shift of functionality from resources to brokers or clients, allowing the resources to be accessed in a standard way and easing the task of sharing resources between organizations and projects. We should consider that the Grid not only involves the technical challenge of constructing and deploying this vast infrastructure, it also brings up other issues related to security and resource sharing policies [4] as well as other socio-political difficulties [5].

Practically, the majority of the Grid infrastructures are being built on protocols and services provided by the Globus Toolkit[1], becoming a *de facto* standard in Grid computing. Globus architecture follows an hourglass approach, which is indeed an "end-to-end" principle [6]. Therefore, instead of succumbing to the temptation of tailoring the core Grid middleware to our needs (since in such case the resulting infrastructure would be application specific), or homogenizing the underlying resources (since in such case the resulting infrastructure would be a highly distributed cluster), we propose to strictly follow the "end-to-end" principle. Clients should have access to a wide range of resources provided through a limited and standardized set of protocols and interfaces. In the Grid, these are provided by the core Grid middleware, Globus, just as, in the Internet, they are provided through the TCP/IP protocols. Moreover, the "end-to-end" principle reduces the firewall configuration to a minimum, which is also welcome by the security administrators.

One of the most ambitious projects to date is EGEE[2] (Enabling Grids for E-sciencE), which is creating a production-level Grid infrastructure providing a level of performance and reliability never achieved before. EGEE currently uses the LCG[3] (LHC Computing Grid) middleware, which is based on Globus. Other much more modest project is IRISGrid[4] (the Spanish Grid Initiative), whose main objective is the creation of a stable national Grid infrastructure. The first version of the IRISGrid testbed is based only on Globus services, and it has been widely used through the Grid*W*ay framework[5].

For the purposes of this paper we have used a Globus-based testbed to run a Bioinformatics application through the Grid*W*ay framework. This testbed was

[1] http://www.globus.org
[2] http://www.eu-egee.org
[3] http://lcg.web.cern.ch
[4] http://www.irisgrid.es
[5] http://www.gridway.org

built up from resources inside IRISGrid and EGEE projects. The aim of this paper is to demonstrate the application of an "end-to-end" principle in a Grid infrastructure, and the feasibility of building loosely-coupled Grid environments based only on Globus services, while obtaining non trivial levels of quality of service through an appropriate user-level Grid middleware.

The structure of the paper follows the layered structure of Grid systems, from bottom-up. The Grid fabric is described Section 2. Section 3 describes the core Grid middleware. Section 4 introduces the functionality and characteristics of the Grid*W*ay framework, used as user-level Grid middleware. Section 5 describes the target application. Finally, Section 6 presents the experimental results and Section 7 ends up with some conclusions.

2 Grid Fabric: IRISGrid and EGEE Resources

This work has been possible thanks to the collaboration of those research centers and universities that temporarily shared some of their resources in order to set up a geographically distributed testbed. The testbed results in a very

Table 1. IRISGrid and EGEE resources contributed to the experiment

Testbed	Site	Resource	Processor	Speed	Nodes	RM
IRISGrid	RedIRIS	heraclito	Intel Celeron	700MHz	1	Fork
		platon	2×Intel PIII	1.4GHz	1	Fork
		descartes	Intel P4	2.6GHz	1	Fork
		socrates	Intel P4	2.6GHz	1	Fork
	DACYA-UCM	aquila	Intel PIII	700MHz	1	Fork
		cepheus	Intel PIII	600MHz	1	Fork
		cygnus	Intel P4	2.5GHz	1	Fork
		hydrus	Intel P4	2.5GHz	1	Fork
	LCASAT-CAB	babieca	Alpha EV67	450MHz	30	PBS
	CESGA	bw	Intel P4	3.2GHz	80	PBS
	IMEDEA	llucalcari	AMD Athlon	800MHz	4	PBS
	DIF-UM	augusto	4×Intel Xeon*	2.4GHz	1	Fork
		caligula	4×Intel Xeon*	2.4GHz	1	Fork
		claudio	4×Intel Xeon*	2.4GHz	1	Fork
	BIFI-UNIZAR	lxsrv1	Intel P4	3.2GHz	50	SGE
EGEE	LCASAT-CAB	ce00	Intel P4	2.8GHz	8	PBS
	CNB	mallarme	2×Intel Xeon	2.0GHz	8	PBS
	CIEMAT	lcg02	Intel P4	2.8GHz	6	PBS
	FT-UAM	grid003	Intel P4	2.6GHz	49	PBS
	IFCA	gtbcg12	2×Intel PIII	1.3GHz	34	PBS
	IFIC	lcg2ce	AMD Athlon	1.2GHz	117	PBS
	PIC	lcgce02	Intel P4	2.8GHz	69	PBS

* These resources actually present two physical CPUs but they appear as four logical CPUs due to hyper-threading.

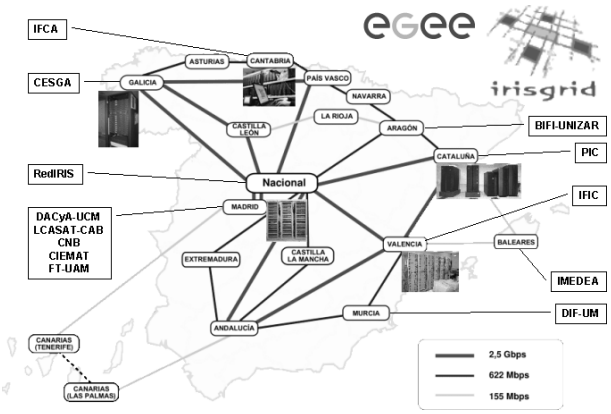


Fig. 1. Geographical distribution and interconnection network of sites

heterogeneous infrastructure, since it presents several middlewares, architectures, processor speeds, resource managers (RM), network links, etc. A brief description of the participating resources is shown in Table 1.

Some centers are inside IRISGrid, which is composed of around 40 research groups from different spanish institutions. Seven sites participated in the experiment by donating a total number of 195 CPUs. Other centers participate in the EGEE project, which is composed of more than 100 contracting and non-contracting partners. Seven spanish centers participated by donating a total number of 333 CPUs.

Together, the testbed is composed of 13 sites (note that LCASAT-CAB is both in IRISGrid and EGEE) and 528 CPUs. In the experiments below, we limited to four the number of jobs simultaneously submitted to the same resource, with the aim of not saturating the whole testbed, so only 64 CPUs were used at the same time. All sites are connected by RedIRIS, the Spanish Research and Academic Network. The geographical location and interconnection links of the different sites are shown in Figure 1.

3 Core Grid Middleware: Globus

Globus services allow secure and transparent access to resources across multiple administrative domains, and serve as building blocks to implement the stages of Grid scheduling [7]. Table 2 summarizes the core Grid middleware components existing in both IRISGrid and EGEE resources used in the experiments. In the case of EGEE, we only used Globus basic services, ignoring any higher-level services, like the resource broker or the replica location service.

Table 2. Core Grid middleware

Component	IRISGrid	EGEE
Security Infrastructure	IRISGrid CA and manually generated `grid-mapfile`	DATAGRID-ES CA and automatically generated `grid-mapfile`
Resource Management	GRAM with shared home directory in clusters	GRAM without shared home directory in clusters
Information Services	IRISGrid GIIS and local GRIS, using the MDS schema	CERN BDII and local GRIS, using the GLUE schema
Data Management	GASS and GridFTP	GASS and GridFTP

We had to introduce some changes in the security infrastructure in order to perform the experiments. For authentication, we used a user certificate issued by DATAGRID-ES CA, so we had to give trust to this CA on IRISGrid resources. Regarding authorization, we had to add an entry for the user in the `grid-mapfile` in both IRISGrid and EGEE resources.

4 User-Level Grid Middleware: Grid *W*ay

User-level middleware is required in the client side to make it easier and more efficient the execution of applications. Such client middleware should provide the end user with portable programming paradigms and common interfaces.

In a Globus-based environment, the user is responsible for manually performing all the submission steps [7] in order to achieve any functionality. To overcome this limitation, Grid *W*ay [8] was designed with a *submit & forget* philosophy in mind. The core of the Grid *W*ay framework is a personal *submission agent* that performs all scheduling stages and watches over the correct and efficient execution of jobs on Globus-based Grids. The Grid *W*ay framework provides adaptive scheduling and execution, as well as fault tolerance capabilities to handle the dynamic Grid characteristics.

A key aspect in order to follow the "end-to-end" principle is how job execution is performed. In EGEE, file transfers are initiated by a job wrapper running in the compute nodes, therefore they act as client machines, so needing network connectivity and client tools to interact with the middleware. In Grid *W*ay, however, job execution is performed in three steps by the following modules:

1. *prolog*: It prepares the remote system by creating a experiment directory and transferring the input files from the client.
2. *wrapper*: It executes the actual job and obtains its exit status code.
3. *epilog*: It finalizes the remote system by transferring the output files back to the client and cleaning up the experiment directory.

This way, GridWay doesn't rely on the underlying middleware to perform preparation and finalization tasks. Moreover, since both *prolog* and *epilog* are submitted to the front-end node of a cluster and *wrapper* is submitted to a compute node, GridWay doesn't require any middleware installation nor network connectivity in the compute nodes.

Other projects [9, 10, 11, 12] have also addressed resource selection, data management, and execution adaptation. We do not claim innovation in these areas, but remark the advantages of our modular, decentralized and "end-to-end" architecture for job adaptation to a dynamic environment.

In this case, we have taken full advantage of the modular architecture of GridWay, as we didn't have to directly modify the source code of the *submission agent*. We extended the *resource selector* in order to understand the GLUE schema used in EGEE. The *wrapper* module also had to be modified in order to perform an explicit file staging between the front-end and the compute nodes in EGEE clusters.

5 Grid Application: Computational Proteomics

One of the main challenges in Computational Biology concerns the analysis of the huge amount of protein sequences provided by genomic projects at an ever increasing pace. In the following experiments, we will consider a Bioinformatics application aimed at predicting the structure and thermodynamic properties of a target protein from its amino acid sequence [13].

The algorithm, tested in the 5th round of Critical Assessment of techniques for protein Structure Prediction (CASP5)[6], aligns with gaps the target sequence with all the 6150 non-redundant structures in the Protein Data Bank (PDB)[7], and evaluates the match between sequence and structure based on a simplified free energy function plus a gap penalty item. The lowest scoring alignment found is regarded as the prediction if it satisfies some quality requirements. In such cases, the algorithm can be used to estimate thermodynamic parameters of the target sequence, such as the folding free energy and the normalized energy gap.

We have applied the algorithm to the prediction of thermodynamic properties of families of orthologous proteins, i.e. proteins performing the same function in different organisms. The biological results of the comparative study of several families of orthologous proteins are presented elsewhere [14].

6 Experiences and Results

The experiments presented here consist in the analysis of a family of 80 orthologous proteins of the *Triose Phosphate Isomerase* enzyme (an enzyme is a special case of protein). Five experiments were conducted in different days during a week. The average turnaround time for the five experiments was 43.37 minutes.

[6] http://PredictionCenter.llnl.gov/casp5/
[7] http://www.pdb.org

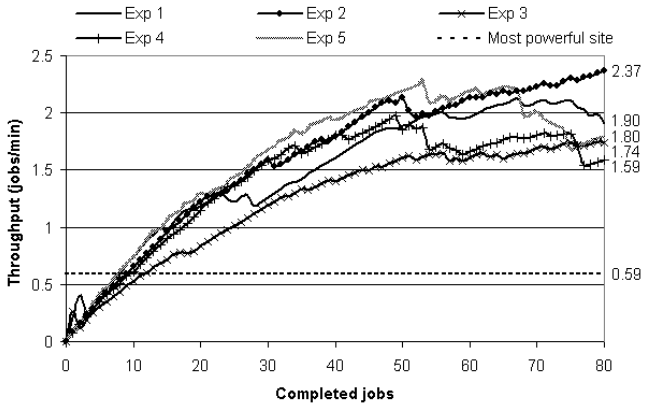


Fig. 2. Testbed dynamic throughput during the five experiments and theoretical throughput of the most powerful site

Figure 2 shows the dynamic throughput achieved during the five experiments alongside the theoretical throughput of the most powerful site, where the problem could be solved in the lowest time, in this case DIF-UM (taking into account that the number of active jobs per resource was limited to four). The throughput achieved on each experiment varies considerably, due to the dynamic availability and load of the testbed. For example, resource ce00 at site LCASAT-CAB was not available during the execution of the first experiment. Moreover, fluctuations in the load of network links and computational resources induced by non-Grid users affected to a lesser extent in the second experiment, as it was performed at midnight.

7 Conclusions

We have shown that the "end-to-end" principle works at the client side (i.e. the user-level Grid middleware) of a Grid infrastructure. Our proposed user-level Grid middleware, Grid*W*ay, can work with Globus, as a standard core Grid middleware, over any Grid fabric in a *loosely-coupled* way. The smooth process of integration of two so different testbeds, although both are based on Globus, demonstrates that the Grid*W*ay approach (i.e. the Grid way), based on a modular, decentralized and "end-to-end" architecture, is appropriate for the Grid.

Moreover, loosely-coupled Grids allow a straightforward resource sharing since resources are accessed and exploited through *de facto* standard protocols and interfaces, similar to the early stages of the Internet. This way, the loosely-coupled model allows an easier, scalable and compatible deployment.

Acknowledgments

We would like to thank all the institutions involved in the IRISGrid initiative and the EGEE project, in particular those who collaborated in the experiments. We would like to also thank Ugo Bastolla, staff scientist in the Bioinformatics Unit at Centro de Astrobiología (CAB) and developer of the Bioinformatics application used in the experiments.

References

1. Foster, I.: What Is the Grid? A Three Point Checklist. GRIDtoday **1**(6) (2002) Available at http://www.gridtoday.com/02/0722/100136.html.
2. Baker, M., Buyya, R., Laforenza, D.: Grids and Grid Technologies for Wide-Area Distributed Computing. Software – Practice and Experience **32**(15) (2002) 1437–1466
3. Allan, R.J., Gordon, J., McNab, A., Newhouse, S., Parker, M.: Building Overlapping Grids. Technical report, University of Cambridge (2003)
4. San José, O., Suárez, L.M., Huedo, E., Montero, R.S., Llorente, I.M.: Resource Performance Management on Computational Grids. In: Proc. 2nd Intl. Symp. Parallel and Distributed Computing (ISPDC 2003), IEEE CS (2003) 215–221
5. Schopf, J.M., Nitzberg, B.: Grids: The Top Ten Questions. Scientific Programming, special issue on Grid Computing **10**(2) (2002) 103–111
6. B. Carpenter, E.: RFC 1958: Architectural Principles of the Internet (1996)
7. Schopf, J.M.: Ten Actions when Superscheduling. Technical Report GFD-I.4, Scheduling Working Group – The Global Grid Forum (2001)
8. Huedo, E., Montero, R.S., Llorente, I.M.: A Framework for Adaptive Execution on Grids. Intl. J. Software – Practice and Experience (SPE) **34**(7) (2004) 631–651
9. Berman, F., Wolski, R., Casanova, H., et al.: Adaptive Computing on the Grid Using AppLeS. IEEE Trans. Parallel and Distributed Systems **14**(4) (2003) 369–382
10. Buyya, R., D.Abramson, Giddy, J.: A Computational Economy for Grid Computing and Its Implementation in the Nimrod-G Resource Broker. Future Generation Computer Systems **18**(8) (2002) 1061–1074
11. Frey, J., Tannenbaum, T., Livny, M., Foster, I., Tuecke, S.: Condor/G: A Computation Management Agent for Multi-Institutional Grids. Cluster Computing **5**(3) (2002) 237–246
12. Vadhiyar, S., Dongarra, J.: A Performance Oriented Migration Framework for the Grid. In: Proc. 3rd Intl. Symp. Cluster Computing and the Grid (CCGrid 2003), IEEE CS (2003) 130–137
13. Huedo, E., Bastolla, U., Montero, R.S., Llorente, I.M.: A Framework for Protein Structure Prediction on the Grid. New Generation Computing **23**(4) (2005) 277–290
14. Bastolla, U., Moya, A., Viguera, E., van Ham, R.: Genomic Determinants of Protein Folding Thermodynamics in Prokaryotic Organisms. Journal of Molecular Biology **343**(5) (2004) 1451–1466

A.4. A Comparative Analysis between EGEE and GridWay Workload Management Systems

Cita Completa

J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *A Comparative Analysis between EGEE and GridWay Workload Management Systems*. International Conference on Grid computing, high-performAnce and Distributed Applications on the Move Federated Conferences (GADA) 2006, Montpellier (Francia), Noviembre 2006. Lecture Notes in Computer Science (LNCS). Vol. 4276, pp. 1143-1151, 2006. Springer Verlag.

Abstract

Metascheduling is a key functionality of the grid middleware in order to achieve a reasonable degree of performance and reliability, given the changing conditions of the computing environment. In this contribution a comparative analysis between two major grid scheduling philosophies is shown: a semi-centralized approach, represented by the EGEE Workload Management System, and a fully distributed approach, represented by the GridWay Metascheduler. This comparative is both theoretical, through a functionality checklist, and experimental, through the execution of a fusion plasma application on the EGEE infrastructure.

Referencia de Citas Bibliográficas

Yu y Buyya (2005); Buyya et al. (2000); Frey et al. (2001); Dail et al. (2004), Berman et al. (2003); Huedo et al. (2004); Vázquez-Poletti et al. (2006); Llorente et al. (2005); Campana et al. (2004); Huedo et al. (2005); Avelino et al. (2004); Morajko et al. (2005); Huedo et al. (2006); Herrera et al. (2004); Castejón et al. (2004); Montero et al. (2006)

Tasa de Aceptación

Este congreso recibió 68 contribuciones, de los cuales se aceptaron 32 (un 47 %).

A Comparative Analysis Between EGEE and Grid *W*ay Workload Management Systems*

J.L. Vázquez-Poletti, E. Huedo, R.S. Montero, and I.M. Llorente

Departamento de Arquitectura de Computadores y Automática. Facultad de Informática, Universidad Complutense de Madrid. 28040 Madrid, Spain

Abstract. Metascheduling is a key functionality of the grid middleware in order to achieve a reasonable degree of performance and reliability, given the changing conditions of the computing environment. In this contribution a comparative analysis between two major grid scheduling philosophies is shown: a semi-centralized approach, represented by the EGEE Workload Management System, and a fully distributed approach, represented by the Grid *W*ay Metascheduler. This comparative is both theoretical, through a functionality checklist, and experimental, through the execution of a fusion plasma application on the EGEE infrastructure.

1 Introduction

The growing computational needs of nowadays projects have permitted the evolution to a new paradigm called Grid Computing. Among the intervening elements of a computional grid, the Metascheduler is gathering most attention as a way to meet these challenging needs. The term Metascheduler can be defined as a grid middleware that discovers, evaluates and allocates resources for grid jobs by coordinating activities between multiple heterogeneous schedulers that operate at local or cluster level [1].

Several implementations of the Metascheduler can be found, such as CSF, Silver, Nimrod/G, Condor-G, GHS, GrADS, MARS, AppLeS and Gridbus. From these, we would like to remark the following: CSF supports advance reservation booking and offers round-robin and reservation based scheduling algorithms. Scheduling characteristics provided by Nimrod/G strive for the equilibrium between resource providers and resources consumers via auctioning mechanisms [2]. Condor-G is not conceived for supporting scheduling policies but in the other hand, it supplies mechanisms, such as *ClassAd* and *DAGMan*, that may be useful for a metascheduler standing above [3]. GrADS and AppLeS support scheduling mechanisms that take into consideration both application and system level environments [4,5]. These solutions provide complementary functionality,

* This research was supported by Consejería de Educación de la Comunidad de Madrid, Fondo Europeo de Desarrollo Regional (FEDER) and Fondo Social Europeo (FSE), through BIOGRIDNET Research Program S-0505/TIC/000101, and by Ministerio de Educación y Ciencia, through the research grant TIC2003-01321. The authors participate in the EGEE project, funded by the European Union.

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4276, pp. 1143–1151, 2006.

which focuses on specific application domains and grid middlewares, contributing significant improvement in the field.

The aim of the Grid*W*ay metascheduler [6] is to provide Globus user community with a grid scheduling functionality similar to that found in local DRM (Distributed Resource Management) systems. In a precedent contribution, we showed how a joint testbed of EGEE (LCG-2 middleware) and non-EGEE resources (Globus middleware) can be harnessed by Grid*W*ay with good results [7]. The evaluation of the resulting infrastructure showed that reasonable levels of performance and reliability were achieved. The objective of this contribution is to compare two metascheduling philosophies by means of two representative implementations: The EGEE WMS (Workload Management System) and Grid*W*ay Metascheduler. In both cases, the experimental results are obtained on EGEE computing resources.

The structure of this paper is as follows. In Section 2, a short architecture overview of the EGEE WMS and the Grid*W*ay Metascheduler can be found, along with a comparative analysis of the functionality provided by both solutions is shown. Then, Section 3 evaluates the performance obtained by the two alternatives in the execution of fusion plasma application on the EGEE infrastructure. Finally in Section 4, the paper ends up with some conclusions.

2 Grid Scheduling Infrastructures

The EGEE WMS and Grid*W*ay are representative metascheduling technologies of different strategies to deploy a grid scheduling infrastructure. Grid*W*ay follows the *loosely-coupled* Grid principle [8], mainly characterized by: autonomy of the multiple administration domains, heterogeneity, scalability and dynamism. A Grid*W*ay instance is installed in each organization involved in the partner grid to provide scheduling functionality for intra-organization users (site-level scheduling). On the other hand, the EGEE WMS provides a higher centralized strategy as one or more scheduling instances can be installed in the grid infrastructure, each providing scheduling functionality for a number of VOs (VO-level scheduling).

The EGEE WMS is the result of previous projects (Datagrid, CrossGrid). The present version of the EGEE WMS is based on the LCG-2 middleware and it is migrating to a new one called gLite which inherits many of the former elements and functionalities. The EGEE WMS architecture is highly centralized and each functionality is provided by almost a different machine, as it is conceived as a network service. The EGEE WMS components are the following: The User Interface (UI), which is from where the user sends the jobs; the Resource Broker (RB), which is based in Condor-G and uses most of its functionality; the Computing Element (CE) and the Worker Nodes (WN), which are the cluster frontend and the cluster nodes respectively, as it is established in the fixed configuration dictated by the EGEE WMS architecture; the Storage Element (SE), which is used for job files storage; the Logging and Bookkeeping service (LB), which registers job events [9].

For this contribution's purpose, we have studied the RB, where the user submits the job and its matching is performed. Here, the RB can adopt an eager or lazy policy for scheduling the jobs. While with the eager policy the job will likely end up in a queue, with lazy scheduling the job is held until a resource becomes available. Then the job is sent to the chosen CE and, from there, executed in a corresponding WN. In any case, there are several alternative brokering services to submit the job.

GridWay works on top of Globus services, performing job execution management and resource brokering, allowing unattended, reliable, and efficient execution of jobs, array jobs, or complex jobs on heterogeneous, dynamic and *loosely-coupled* Grids formed by Globus resources. GridWay's modular architecture is conformed by the GridWay Daemon (GWD) and different Middleware Access Drivers (MADs) to access different Grid Services (information, execution and transfer), all of them in just one host, as GridWay is conceived as a client tool. GridWay performs all the job scheduling (using a lazy approach) and submission steps transparently to the end user adopting job execution to changing Grid conditions by providing fault recovery mechanisms, dynamic scheduling, migration on-request and opportunistic migration. GridWay allows the deployment of organization-level meta-schedulers that provide support for multiple intra-organization users in each scheduling instance. There is one scheduling instance for each organization and all instances compete with each other for the available resources.

2.1 Scheduling Capabilities

Both EGEE WMS and GridWay treat jobs in a FIFO way and support dynamic scheduling, providing a way to filter and evaluate resources based on dynamic attributes, by means of requirement and rank expressions. In the EGEE WMS, the names of these attributes are the same as retrieved from the information service. Nevertheless, in GridWay, these expressions are based on common resource attributes, independent from the information service, providing another way of decoupling. While the EGEE WMS RB accesses only the BDII servers and only processes the GLUE Schema, GridWay's different information MADs allow to access the most common information services. Considering execution and transfer functionalities, both EGEE WMS and GridWay support Globus Pre-WS services, but only GridWay allows access to Globus WS services [10].

GridWay supports opportunistic migration. That is, each scheduling cycle evaluates the benefits of migrating submitted jobs to new available resources (recently added or freed) by comparing rank values. In the EGEE WMS, this functionality is not supported and the ranking only affects submission.

Considering performance slowdown detection, GridWay takes count of the suspension time in remote batch systems and requests a migration when it exceeds a given threshold. Moreover, jobs are submitted together with a lightweight self monitoring system. The job will migrate when it doesn't receive as much CPU as the user expected. None of the performance slowdown detection mechanisms given by GridWay are implemented in the EGEE WMS. The

monitoring in the EGEE WMS architecture is provided by the LB, which records only basic job states and mixes them with events originated in other components.

With Grid*W*ay, an application can take decisions about resource selection as its execution evolves by modifying its requirement and rank expressions and requesting a migration. In the EGEE WMS RB instead, these expressions are only set at the beginning.

The EGEE WMS supports checkpointing by providing an API to allow applications to be instrumented to save the state of the process (represented by a list of variable and value pairs) at any moment during the execution of a job. Also, it provides the mechanisms to restart the computation from checkpointed data (previously saved state). If a job fails, the WMS automatically reschedules the job and resubmits it to another compatible resource. There, the last state is automatically retrieved and the computation is restarted. The user can also retrieve the saved state for a later manual resubmission, where the user can specify if the job must start from this retrieved checkpoint data. With Grid*W*ay, user-level checkpointing or architecture independent restart files managed by the programmer must be implemented. Migration is implemented by restarting the job on the new candidate host. If the checkpointing files are not provided, the job should be restarted from the beginning. These checkpoints are periodically retrieved to the client machine or a checkpoint server.

In the EGEE WMS, the PBS Event Logging (APEL) is employed for distributed accounting [11]. Grid*W*ay gives the user local accounting functionalities, standing on the Berkeley Database.

Dependency in job submission is supported in the EGEE WMS RB by the Condor's DAGMan tool [11]. A DAGMan process is locally spawned for each Direct Acyclic Graph (DAG) submitted to Condor-G. Grid*W*ay also provides support for job dependencies.

2.2 Fault Detection and Recovery Capabilities

The EGEE WMS RB incorporates error detection mechanisms provided by Condor-G [12]. On the other hand, Grid*W*ay detects job cancellation (when the job exit code is not specified), remote system crash and network disconnection (both when the polling of the job fails). In all of these cases, Grid*W*ay requests a migration for the job [13].

Also the system running the scheduler could fail. Grid*W*ay persistently saves its state in order to recover or restart the jobs when the system is restarted. The EGEE WMS RB relies on Condor-G, which stores persistently all relevant state for each submitted job in the scheduler's job queue [3].

2.3 User Interface Functionality

Both Grid*W*ay and the EGEE WMS RB allow single jobs. The EGEE WMS RB can handle jobs with dependencies (DAGMan functionality) and interactive jobs. On the other hand, Grid*W*ay allows array jobs, jobs with dependencies and complex jobs. For providing complex job support, Grid*W*ay gives the user

a functionality to synchronize jobs. In the case of the EGEE WMS, this must be implemented by a periodical polling (active wait).

Focusing in the command line interface, both Grid*W*ay and the EGEE WMS RB give the user full control of his jobs. Anyway, Grid*W*ay incorporates commands which allow the user to migrate and synchronize jobs, functionalities not provided by the EGEE WMS.

Also, Grid*W*ay offers C and Java implementations of the DRMAA Application Programming Interface, which is a Global Grid Forum (GGF) standard [14]. The EDG WMS API given by the EGEE WMS[1] is not standard.

3 Experimental Conditions and Results

The grid infrastructure used for the experiments is the corresponding to the Test Virtual Organization at the Southwest Federation (SWETEST VO) of the EGEE project (Table 1). All Spanish sites are connected by RedIRIS, the Spanish Research and Academic Network, whose interconnection links of the different nodes are shown in Figure 1.

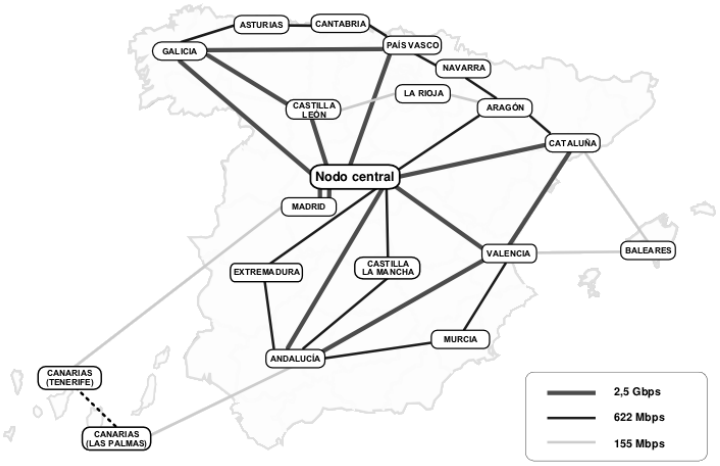


Fig. 1. Topology and bandwidths of RedIRIS-2

The target application, called *Truba*, performs the tracing of a single ray of a microwave beam launched inside a fusion reactor [15]. Each experiment involves the execution of 50 instances of the *Truba* application. The experiments were performed with a development version of *Truba*, whose average execution time

[1] http://www.to.infn.it/grid/workload_management/apiDoc/edg-wms-api-index.html

on a Pentium 4 3.20 GHz is 9 minutes. *Truba*'s executable file size is 1.8 MB, input file size is 70 KB, and output file size is about 549 KB.

For the EGEE WMS experiments, we have developed a framework using the lcg2.1.69 User Interface C++ API, which provides support to submit, monitor and control each single ray tracing application to the grid. This framework works in the following way: First of all, a launcher script generates the JDL files needed. Then, the framework launches all the single ray tracing jobs simultaneously, periodically querying each job's state. And finally, it retrieves the job's output. The scheduling decisions are of course delegated to the EGEE WMS.

Grid*Way* only relies on Globus services, so it could be used in any Grid infrastructure based on the Globus Toolkit, both PreWS and WS [10]. In the case of the EGEE WMS (LCG-2), Globus behaviour has been slightly modified, but it does not loose its main protocols and interfaces, so Grid*Way* can be used in a standard way to access LCG-2 resources [7].

Table 1. EGEE grid resources employed during the experiment

Site	Processor	Speed	Nodes	DRMS
CESGA	Intel Pentium III	1.1 GHz	46	PBS
IFAE	Intel Pentium 4	2.8 GHz	11	PBS
IFIC	AMD Athlon	1.2 GHz	127	PBS
INTA-CAB	Intel Pentium 4	2.8 GHz	4	PBS
LIP	Intel Xeon	2.8 GHz	25	PBS
PIC	Intel Pentium 4	2.8 GHz	172	PBS
USC	Intel Pentium III	1.1 GHz	100	PBS

In both cases, the jobs were submitted from Universidad Complutense de Madrid. The RB employed for the experiments with the EGEE WMS was located at the IFIC site and used an eager scheduling policy.

3.1 Experimental Results

Table 2 shows a summary of the performance exhibited by the two scheduling systems in the execution of the fusion application. As can be seen, Grid*Way* presents a higher transfer time, because of the reverse-server transferring model used for file staging [6] (which has been replaced in version 4.7 for solving this issue). Moreover, the standard deviation of raw performance metrics can be interpreted as an indicator of the heterogeneity in the grid resources and interconnection links [7]. Finally, the lower overhead induced by Grid*Way* shows the benefits of its lighter approach and the functionality for performance slowdown detection.

The EGEE WMS spent 195 minutes (3.25 hours) to execute the 50 jobs, giving a productivity equal to 15.38 jobs/hour. Grid*Way* spent 120 minutes (2 hours) to execute the same workload, giving a productivity equal to 25 jobs/hour. We can conclude that Grid*Way* takes better advantage of the available resources due to its superior scheduling capabilities on dynamic resources. In fact, during the experiments with the EGEE WMS, several problems described before were

Table 2. Performance metrics for both platforms, times are in minutes and productivity is in jobs/hour

	Execution/Job		Transfer/Job		Turnaround	Productivity	Overhead/Job
Framework	Mean	Dev.	Mean	Dev.			
EGEE WMS	30.33	11.38	0.42	0.06	195	15.38	1.82
Grid*W*ay	36.80	16.23	0.87	0.51	120	25.00	0.52

evidenced. The EGEE WMS RB does not provide support for opportunistic migration and slowdown detection, and jobs are assigned to busy resources.

Additionally, the achieved level of parallelism [16] can be obtained by using the following expression:

$$U = \frac{T_{exe}}{T}, \tag{1}$$

being T_{exe} the sum of job execution times and T the turnaround time. The level of parallelism achieved by Grid*W*ay was higher than the level achieved by the EGEE WMS (14.91 and 6.89 respectively).

Not all jobs ended successfully at the first try. In the case of the EGEE WMS, 31 jobs were affected and they had to be resubmitted. However, with Grid*W*ay, only 1 job failed, but there were 21 migrations mostly due to suspension timeouts (too much delay in a queue), and better resource discovery (too much time allocated to a resource when better resources are waiting to be used).

A methodology to analyze the performance of computational Grids in the execution of high throughput computing applications has been proposed in [17]. This performance model enables the comparison of different platforms in terms of the following parameters: asymptotic performance (r_∞), which is the maximum rate of performance in tasks executed per second, and half-performance length ($n_{1/2}$), which is the number of tasks required to obtain half of the asymptotic performance. A first order characterization of a grid by means of these parameters is:

$$n(t) = r_\infty t - n_{1/2}. \tag{2}$$

Then, we can define the performance of the system, jobs completed per second, with a finite number of tasks with:

$$r(n) = n(t)/t = \frac{r_\infty}{1 + n_{1/2}/n}, \tag{3}$$

where n is the number of jobs. The parameters of the model, r_∞ and $n_{1/2}$, are obtained by linear fitting to the experimental results obtained in the execution of the applications.

Figure 2 and Figure 3 show the experimental performance obtained with the two workload management systems, along with that predicted by Eq. (2) and Eq. (3). With the EGEE WMS, r_∞ was 0.0051 jobs/second (18.19 jobs/hour) and $n_{1/2}$ was 8.33. With Grid*W*ay, r_∞ was 0.0079 jobs/second (28.26 jobs/hour) and $n_{1/2}$ was 1.92. From the different values of $n_{1/2}$, we can deduce that Grid*W*ay needs less jobs to obtain half of the asymptotic performance due to an earlier job allocation in the resources.

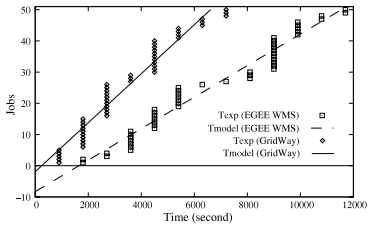


Fig. 2. Measurements of r_∞ and $n_{1/2}$ parameters for both platforms

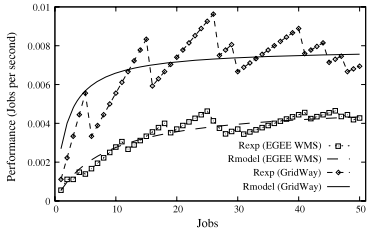


Fig. 3. Experimental and predicted performance

4 Conclusions

We have demonstrated that Grid*Way* achieves lower overhead and higher productivity than the EGEE WMS. Grid*Way* reduces the number of job submission stages and provides mechanisms, not given by the EGEE WMS RB, such as opportunistic migration and performance slowdown detection that considerably improves the usage of the resources. Nevertheless, EGEE WMS provides other components that weren't considered in this article, such as data management.

Acknowledgments

We would like to thank all the institutions involved in the EGEE project, in particular those who collaborated in the experiments.

References

1. Yu, J., Buyya, R.: A Taxonomy of Workflow Management Systems for Grid Computing. Journal of Grid Computing **3** (2005) 171–200
2. Buyya, R., Abramson, D., Giddy, J.: Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid. Fourth International Conference on High-Performance Computing in the Asia-Pacific Region **1** (2000) 283

3. Frey, J., Tannenbaum, T., Livny, M., Foster, I., Tuecke, S.: Condor-G: A Computation Management Agent for Multi-Institutional Grids. In: HPDC '01: Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10'01), IEEE Computer Society (2001) 55
4. Dail, H., Sievert, O., Berman, F., Casanova, H., YarKhan, A., Vadhiyar, S., Dongarra, J., Liu, C., Yang, L., Angulo, D., Foster, I.: Scheduling in the Grid Application Development Software Project. In: Grid resource management: state of the art and future trends. Kluwer Academic Publishers (2004) 73–98
5. Berman, F., Wolski, R., Casanova, H., Cirne, W., Dail, H., Faerman, M., Figueira, S., Hayes, J., Obertelli, G., Schopf, J., Shao, G., Smallen, S., Spring, N., Su, A., Zagorodnov, D.: Adaptive Computing on the Grid Using AppLeS. IEEE Transactions on Parallel and Distributed Systems **14** (2003) 369–382
6. Huedo, E., Montero, R.S., Llorente, I.M.: A Framework for Adaptive Execution on Grids. Intl. J. Software – Practice and Experience (SPE) **34** (2004) 631–651
7. Vázquez-Poletti, J., Montero, R.S., Llorente, I.M.: Coordinated Harnessing of the IRISGrid and EGEE Testbeds with GridWay. Journal of Parallel and Distributed Computing **66** (2006) 763–771
8. Llorente, I.M., Montero, R.S., Huedo, E.: A Loosely Coupled Vision for Computational Grids. IEEE Distributed Systems Online **6** (2005)
9. Campana, S., Litmaath, M., Sciaba, A.: LCG-2 Middleware Overview. Available at https://edms.cern.ch/document/498079/0.1 (2004)
10. Huedo, E., Montero, R.S., Llorente, I.M.: Coordinated Use of Globus Pre-WS and WS Resource Management Services with GridWay. In: Proc. 2nd Workshop on Grid Computing and its Application to Data Analysis (GADA'05) on the Move Federated Conferences. Volume 3762 of Lecture Notes in Computer Science. (2005) 234–243
11. Avellino, G., Beco, S., Cantalupo, B., et al: The DataGrid Workload Management System: Challenges and Results. Journal of Grid Computing **2** (2004) 353–367
12. Morajko, A., Fernandez, E., Fernandez, A., Heymann, E., Senar, M.A.: Workflow Management in the CrossGrid Project. In: Proc. European Grid Conference (EGC2005). Volume 3470 of Lecture Notes in Computer Science. (2005) 424–433
13. Huedo, E., Montero, R.S., Llorente, I.M.: Evaluating the Reliability of Computational Grids from the End User's Point of View. Journal of Systems Architecture, (2006) (in press).
14. Herrera, J., Montero, R., Huedo, E., Llorente, I.: DRMAA Implementation within the GridWay Framework. In: Workshop on Grid Application Programming Interfaces, 12th Global Grid Forum (GGF12). (2004)
15. Castejon, F., Tereshcenko, M.A., et al.: Electron Bernstein Wave Heating Calculations for TJ-II Plasmas. American Nuclear Society **46** (2004) 327–334
16. Huedo, E., Montero, R.S., Llorente, I.M.: An Evaluation Methodology for Computational Grids. In: Proc. 2005 International Conference on High Performance Computing and Communications. Volume 3726 of Lecture Notes in Computer Science. (2005) 499–504
17. Montero, R.S., Huedo, E., Llorente, I.M.: Benchmarking of High Throughput Computing Applications on Grids. Parallel Computing (2006) (in press).

A.5. Fusion in the Grid

Cita Completa

F. Castejón, I. Campos, A. Cappa, L.A. Fernández, E. Huedo, I.M. Llorente, V. Martín, R.S. Montero, M. Mikhailov, A. Tarancón, M.A. Tereschenko, J.L. Vázquez-Poletti, J.L. Velasco, V. Voznesensky. *Fusion in the Grid*. Spanish Conference on e-Science Grid Computing, CIEMAT Madrid (España), Marzo 2007.

Abstract

In this paper, a report is given on the status of the fusion applications in grid infrastructures. Several pilot applications are already running and have produced relevant scientific results. Some notions on the role of computing science to solve the open problems that are remaining in Fusion and Plasma Physics are also given. Grid infrastructures together with supercomputers are the suitable tools for solving such problems depending on their structure.

Referencia de Citas Bibliográficas

Nüremberg et al. (2005); Castejón y Eguilior (2003); Vargas et al. (2007); Cappa et al. (2005); Vázquez-Poletti et al. (2006); Spong (2005); Spong (2006); Castejón et al. (2006)

Notas Adicionales

La importancia de esta publicación radica en el impacto que ha tenido el trabajo relativo a esta Tesis Doctoral en áreas científicas fuera del ámbito de la Computación.

Fusion in the Grid

Presenting author:
Francisco Castejón
francisco.castejon@ciemat.es

Laboratorio Nacional de Fusión. Asociación EURATOM-CIEMAT, Madrid, Spain.

Coauthors:

I. Campos[2], A. Cappa[1], L. A. Fernández[3,7], E. Huedo[4], I.M. Llorente[4], V. Martín[3,7], R.S. Montero[4], M. Mikhailov[5], A. Tarancón[7], M.A. Tereshchenko[6], J.L. Vazquez-Poletti[4], J. L. Velasco[7], V. Voznesensky[5]

1) Laboratorio Nacional de Fusión. Asociación EURATOM/CIEMAT para Fusión. 28040, Madrid, Spain
2) Instituto de Física de Cantabria. CSIC. 39005, Santander, Spain
3) Facultad de Físicas. Universidad Complutense de Madrid, 28040, Madrid, Spain
4) Facultad de Informática. Universidad Complutense de Madrid, 28040, Madrid, Spain.
5) Insitute Kurchatov. Moscow, Russia.
6) Institute of General Physics. Moscow. Russia.
7) BIFI: Instituto de Biocomputación y Física de Sistemas Complejos. University of Zaragoza, 50006, Zaragoza, Spain

Abstract
In this paper, a report is given on the status of the fusion applications in grid infrastructures. Several pilot applications are already running and have produced relevant scientific results. Some notions on the role of computing science to solve the open problems that are remaining in Fusion and Plasma Physics are also given. Grid infrastructures together with supercomputers are the suitable tools for solving such problems depending on their structure.

1) Motivation: Computing in Plasma Physics.

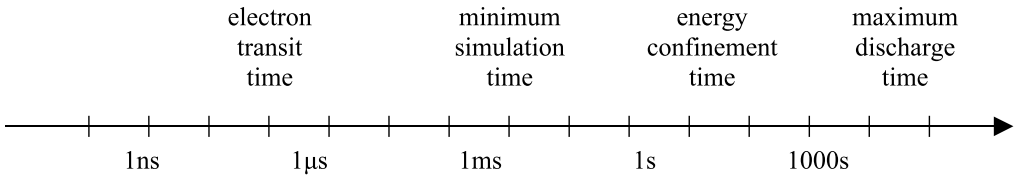
Plasmas are complex systems, with a lot of non-linear processes, with all the time and space scales playing a role (selfsimilarities and self-organization processes are at work). The research in Plasma Theory can therefore help to advance in chaos and self-organisation theory. And viceversa, this kind of disciplines can provide tools for Plasma Physics and Fusion research.

Moreover, plasmas are out of the equilibrium, open systems, which imply the appearance of challenges for Statistical Physics. The development of Thermodynamics of system outside the equilibrium can be helped by Plasma Physics research.

Plasmas are also in the intermediate range between Fluid and Kinetic Theories. Both of them are at work at given ranges of plasma parameters.

Therefore, beyond the obvious interest that Plasma Physics has for commercial Fusion development, it is connected with some of the present challenging disciplines in Physics. And there are still open problems whose solutions can help in commercial fusion achievement.

Due to the complex nature of these open problems, computing is a key element for solving them and, moreover, Computational Plasma Physics can be a motivation for developing powerful computers. For full plasma simulations that take into account everything in the plasma, these are the time scales and run estimates:



Today, one could perform a 1 ms simulation within several days on a 50 - 100 TFlop/s machine like Mare Nostrum[1]. In 2014, a transport-time-scale (i. e., in the energy confinement time scale) simulation will take several weeks on a 4 PFlop/s machine, assuming that computer power doubles every 18 months.

1.1) Stellarators & Tokamaks

The most promising magnetic confinement concept to build a reactor are stellarators and tokamaks, which present different Physical properties and challenges from the computation point of view.

Stellarators are fully 3D systems with a complicated geometry that must be considered exactly in almost all the problems. Usually, this fact implies the necessity of performing heavy calculations. The number of Fourier harmonics to describe a single closed magnetic surface can vary from 50 to 150 which makes that computation in stellarators can be especially heavy.

Tokamaks are usually considered as 2D devices, but there are some specific problems that force one to take into account the real 3D geometry: Those problems in which the real geometry of vacuum vessel has to be taken into account or the cases in which the toroidal ripple due to the finite number of coils must be considered are examples of 3D problems in tokamaks.

2) Parallel vs. Distributed problems in Fusion.

Similarly to other disciplines, two main kinds of problems can be considered in computational Plasma Physics, depending on the structure of the problems and on the interaction between their elements. The different computer architectures can be more or less suitable depending on the type of problem to solve:

Parallel problems are those that require a lot of communication between processors to be solved. The collective behaviour is dominant. Examples: Fluid Theory. Gyrokinetic codes. Turbulence. Equilibrium and Stability… Massively Parallel supercomputers with share memory or fast communication between processors are suitable tools for studying this type of phenomena.
Distributed computation problems: a large fraction of the problem can be treated solving for its independent elements. Some examples: Kinetic theory. Massively repeated calculations. Monte Carlo codes…

3) Grids and distributed problems in Fusion.

It is possible to find a number of problems in Plasma Physics that need distributed calculations to be solved. These are solved by codes suitable to run in the Grid, which provides large cheap computing capability. Some examples are listed here:

- Monte Carlo codes:
    - Plasma-wall interaction; neutral particle orbits.
    - Kinetic transport: guiding centre orbits in tokamaks and stellarators.
- Langevin equations problem that needs the separate estimation of every single particle[2].
- Massive Transport analysis. An example of massive Transport analysis performed in an non-automtic way can be found in [3].
- Massive Ray Tracing[4].
- Stellarator Optimization[5].

EGEE (Enabling Grids for E-sciencE) Project provides durable Infrastructure suitable to perform large calculations in the grid. A Fusion Virtual Organization (Fusion VO) is now working devoted to Fusion calculations, with partners from Spain, Russia, U.K., Italy, France, and Korea[6].

For the moment this is a pilot experience to explore the utility of the Grid for fusion problems. Apart from the above mentioned infrastructure, this experience takes advantage of the fact that EGEE provides Middleware suitable for launching works to the grid and to manage data (for instance gLite).

4) Presently running applications:

The three applications that were chosen as pilot ones are now running in the grid and producing relevant scientific results. These applications were chosen because of their simple workflow and because all the involved calculations are distribute3d ones:

4.1) Kinetic Transport

The idea is to estimate transport properties of Fuison devices by following independent particle orbits in the plasma, according to the well known guiding centre equation:

$$\vec{V}_D = \frac{\vec{E} \times \vec{B}}{B^2} + \frac{\mu}{2\theta}(2\vec{\omega} - \vec{\omega}_{\parallel}) \frac{\vec{B} \times \nabla \vec{B}}{B^3}$$

As a first step we solved these equations in TJ-II stellarator, which has a very complicated geometry, without collisions[7]. The effect of collisions have been included by adding a stochastic term[8]. Typically 10^7 ions must be followed to obtain representative results and Montecarlo techniques are used: Particles are distributed randomly, according to the experimental density and ion temperature profiles (Maxwellian distribution function. The next step could be to add Langevin Equations for heating and turbulence.

An example of orbit in the real 3D Stellarator geometry takes 10 s CPU time in a single CPU. Total 10^7 s. the total distribution function can be obtained at a given position ~1 GBy data, 24 h x 512 CPUs.

4.4) Multiple Ray Tracing Calculations.

The microwave beam for plasma heating can be simulated by a bunch of rays with different wave numbers. A single weakly relativistic ray takes about 10 min in a 3D plasma confinement device [9].
The microwave beam is simulated by 100-200 rays in the usual situations, but
(100-200 rays) x (100-200 wave numbers) ~10^5 rays can be needed for some cases in Electron Bernstein Wave calculations 4.

4.5) Stellarator optimization in the Grid

A lot of different Stellarators (different Magnetic Configurations) are operating nowadays. The optimization based on the knowledge of stellarator physics is necessary. It can be performed numerically by variation of the field parameters. Magnetic field described by Fourier series and every variant computed on a separate processor (~10') using VMEC (Variational Momentum Equilibrium Code). The outermost magnetic surface can be described by about 120 Fourier modes.
The optimization criteria can be:
- Minimizing Neoclassical Transport and Bootstrap current.
- Equilibrium and plasma stability at high plasma pressure.

A Genetic Algorithm to select the optimum configuration for given Target Functions.

5) Envisaged applications to be ported to the grid.

Beyond the former pilot applications that have been used to obtain relevant scientific results, several problems that involve a large fraction of distributed calculation could be considered to be solved with the help of the grid. It is under study what the most convenient to process with, but a list is given here:

5.1) DKES and Neoclassical calculations.

The Grid can also be useful to estimate the trajectories of ions in the NBI injectors for ITER. Ion trajectories must be estimated for correcting the deflection originated by the residual ITER magnetic field: Ions leave the Ion source extracting grid and move in the magnetic field until they are neutralized at the neutralizer. Compensation coils must be designed and different field scenarios must be considered in the design.

To calculate ion deflections in the magnetic field high statistics Monte Carlo calculations are needed. Single trajectories must be estimated

5.2) Ion trajectories in ITER NBI system.

The Grid can also be useful to estimate the trajectories of ions in the NBI injectors for ITER. Ion trajectories must be estimated for correcting the deflection originated by the residual ITER magnetic field: Ions leave the Ion source extracting grid and move in the magnetic field until they are neutralized at the neutralizer. Compensation coils must be designed and different field scenarios must be considered in the design.

To calculate ion deflections in the magnetic field high statistics Monte Carlo calculations are needed. Single trajectories must be estimated

5.3) Plasma-Wall Interaction

EDGE2D and EIRENE codes are used both for tokamaks & Stellarators. These codes perform their estimates by following a large number of neutral particles in a plasma background. The real Geometry of the wall and all the elements inside the vessel are needed.
The neutrals perform independent orbits. Interaction with a transport code is needed, after estimating the plasma source.

EIRENE Code
Two parts: 1) Following trajectories (Totally distributed) --> GRID
2) Reduction to put all together.
EIRENE Code comes from IPP (Jülich, Germany) and is extensively used by Fusion community.

EDGE2D solves the 2D fluid equations for the conservation of energy, momentum and particles in the plasma edge.
Ions, electrons and all ionisation stages of multiple species are considered.
Interaction with the vessel is simulated by coupling to Monte Carlo codes to provide the neutral ion and impurity sources.

5.3) Massive Transport Calculations

Obtaining the parametric dependence of transport coefficients for a large amount of discharges can be done by using grid techniques. A different case can be run on every CPU.

- **6) Conclusions & Future Needs**

Computational Plasma Physics is a challenging discipline that can push forwards Physics Frontiers by giving new results in some open and first line Physics topics.

The present generation of Supercomputers (~50 Tflops) is overcome by some open problems and this is true in particular for some needed simulations for ITER. Some work on Computational Plasma Physics must be done and more powerful computers are needed for this purpose. It is necessary, in particular, to keep in mind the integration of models to walk towards the Numerical Tokamak and Stellarator, although the latter is more challenging.

There are a lot of relevant problems that need mainly distributed calculations, which make them suitable for running in the Grid that can be considered as a cheap supercomputer. VO of Fusion grid is working and ready to be used by the Plasma Physcis community. It is even possible to think of workflows that involve both supercomputers and grid computing, since there are several applications that could take advantage of the two kinds of computational approaches.

[1] C. Nürenberg et al. “Global ITG Turbulence in screw-pinch geometry”. Proceedings of the 15th Styellarator Workshop and the IAEA-TM on Stelarator Theory and Innovative Concepts.
[2] F. Castejón and S. Eguilior. Plasma Physics and Controlled Fusion 45 (2003) 159.
[3] I. Vargas, D. López Bruna, J. Herranz, and F. Castejón. “Experimental electron heat diffusion in TJ-II ECRH plasmas” To be Publisher in Nuclear Fusion, 2007.
[4] A. Cappa, F. Castejón, E. Holzauer2, M. Tereshchenko and A. Fernández “Optimum Gaussian beam for O-X conversion in EBW heated plasmas” Proceedings of the 11th European Fusion Theory conference.
J. L. Vázquez Poletti. “MaRaTra: An application for Massive Ray Tracing in Plasmas”. Proceedings of EGEE·, 06 Userforum.
[5] D. A. Spong “Physics of Plasmas” 12 (2005) 056114
D. A. Spong. Fusion Science and Technology 50 (2006) 343
[6] http://grid.bifi.unizar.es/egee/fusion-vo//
[7] F. Castejón et al. Fusion Science and Technology 50 (2006) 412.
[8] F. Castejón et al. “Ion Kinetic transport in the presence of collisions and electric field” Submitted to Plasma Physics and Controlled Fusion.
[9] F. Castejón et al. “Relativistic and non-relativistic
Fusion Science and Technology 50 (2006) 412.

A.6. Advanced Strategies for Efficient Workflow Management in a Protein Clustering Application with GridWay

Cita Completa

J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *Advanced Strategies for Efficient Workflow Management in a Protein Clustering Application with GridWay*. 1st Iberian Grid Infrastructure Conference, Santiago de Compostela (España), Mayo 2007. CESGA, pp. 199-207.

Abstract

Bioinformatics is demanding the computational power offered by Grid Computing. On the other hand, a workflow management system is needed for the complex applications pertaining to this research area. In a precedent contribution, a Bioinformatics application which performs protein clustering was ported to the Grid using the GridWay metascheduler on the EGEE production infrastructure. The reason for the Grid approach is that a single machine is subject to memory restrictions and the input database size grows everyday. Nevertheless, execution times obtained on the Grid were higher due to the overheads imposed by its highly dynamic, heterogeneous and faulty environment. In this paper we introduce some optimization strategies that will shorten execution times in the considered workflow.

Referencia de Citas Bibliográficas

Yu y Buyya (2005); Vázquez-Poletti et al. (2007); Li y Godzik (2006); Thain et al. (2003); Deelman et al. (2004); Taylor et al. (2004); Kurowski et al. (2004); McGough et al. (2004); Laszewsi et al. (2004); Yu y Buyya (2004);Huedo et a. (2004); Pinedo (2002)

Notas Adicionales

La ausencia de datos del baremo que evalúa el congreso (perteneciente a la Infraestructura Grid Ibérica) en el que se presentó esta publicación, se debe a que es el primero de su serie.

Advanced Strategies for Efficient Workflow Management with Grid *W*ay*

J.L. Vázquez-Poletti, E. Huedo, R.S. Montero, and I.M. Llorente

Universidad Complutense de Madrid, 28040 Madrid, Spain,
WWW home page: http://asds.dacya.ucm.es/

Abstract. Bioinformatics is demanding the computational power offered by Grid Computing. On the other hand, a workflow management system is needed for the complex applications pertaining to this research area. In a precedent contribution, a Bioinformatics application which perforns protein clustering was ported to the Grid using the Grid *W*ay metascheduler on the EGEE production infrastructure. The reason for the Grid approach is that a single machine is subject to memory restrictions and the input database size grows everyday. Nevertheless, execution times obtained on the Grid were higher due to the overheads imposed by its highly dynamic, heterogeneous and faulty environment. In this paper we introduce some optimization strategies that will shorten execution times in the considered workflow.

1 Introduction

Bioinformatics stands among the different research areas which are profusely using Grid Computing. The complexity of the proposed problems involves the management of workflows. The workflow concept appears to satisfy the need to automate procedures in data transfer and job execution. So, a workflow management system is the one which defines, manages and executes workflows over a Grid [1].

In a previous contribution [2], a protein clustering application called *cd-hit* [3], used by the Spanish National Oncology Research Center (Centro Nacional de Investigaciones Oncológicas - CNIO)[1], was introduced for its porting to the Grid. Several workflow management systems were considered for this

* This research was supported by Consejería de Educación of Comunidad de Madrid, Fondo Europeo de Desarrollo Regional (FEDER) and Fondo Social Europeo (FSE), through BioGridNet Research Program S-0505/TIC/000101, and by Ministerio de Educación y Ciencia, through research grant TIN2006-02806. This work makes use of results produced by the Enabling Grids for E-sciencE project, a project co-funded by the European Commission (under contract number INFSO-RI-031688) through the Sixth Framework Programme. EGEE brings together 91 partners in 32 countries to provide a seamless Grid infrastructure available to the European research community 24 hours a day. Full information is available at http://www.eu-egee.org/.

[1] http://www.cnio.es/

task: The Directed Acyclic Graph Manager (DAGMan) [4] provided by Condor, Pegasus [5], Triana [6] standing on the Grid Application Toolkit (GAT) from GridLab [7], ICENI [8], GridAnt [9], Gridbus [10] and Grid*W*ay [11]. From these, Grid*W*ay was chosen because of its workload management and fault tolerance capabilities. On the other hand, the EGEE infrastructure was employed due to its vastness of resources as it is a production Grid.

Different results were obtained depending on the number of partitions performed to a mid-sized protein database, and a performance study was accomplished. Nevertheless, some lacks were found in the model (how the algorithm was ported onto the Grid), as it is described in Section 2. In this work we present new improvement strategies in order to minimize the execution time (see Section 3). Finally, the conclusions and the indication of future work are presented in Section 4.

2 Previous Results and Model Limitations

The Bioinformatics application called *cd-hit* performs protein clustering, in order to eliminate redundancies in a protein database. Its algorithm is represented in Figure 1 and works as described below. In the beginning, a tool called *cd-hit-div* performs a protein database division. The first division, which is the representative one, is processed by the *cd-hit* tool in order to perform a comparison with itself (represented as the A task in Figure 1). The output is then compared to te rest of partitions through the *cd-hit-2d* tool (represented as the B tasks). The first partition which results from the last operation is at this time the representative one (the A' task), and the process starts over until there aren't any more partitions. Finally, when the last comparison is performed, all the outputs of the *cd-hit* tool are merged with the *clstr_merge.pl* tool.

In the previous work, a mid-sized database with 504,876 proteins (435MB) was processed. This input database pertained to the RefSeq [2] database, provided by the National Center for Biotechnology Information (NCBI). The infrastructure employed for running the experiments belongs to the Enabling Grids for E-sciencE (Table 1).

With the purpose of porting the application to the Grid, tasks are divided in two types. In the first type, the job executes both *cd-hit-2d* and *cd-hit* over the given database division. In the second type, the job executes just *cd-hit-2d*. Both the database division and final merging are locally performed. As it can be understood from Figure 1, tasks from a certain level cannot start until the first type job from the previous level is not finished, so task dependencies must be managed in this workflow. In a workflow, the node path translated into a sequence of tasks, to which the completion time is subject, is called *critical path* [12]. In this case, the *critical path* is composed by the execution of the first type tasks (the shaded nodes in the Figure).

Considering the jobs submitted and in particular their input, both the file size and the number of the resulting tasks depend on the number of database

[2] http://www.ncbi.nlm.nih.gov/RefSeq/

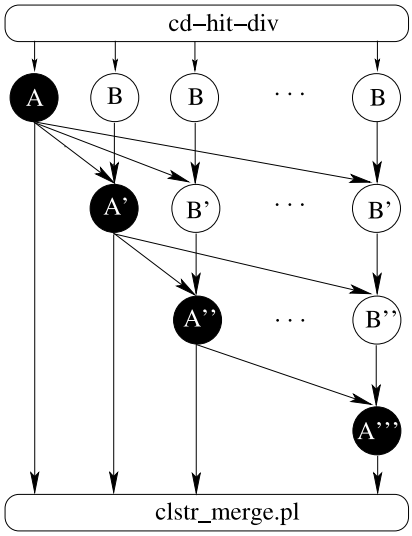


Fig. 1. The *cd-hit* algorithm. White tasks execute *cd-hit-2d* and shaded tasks execute both *cd-hit-2d* and *cd-hit*.

Table 1. Testbed resources. All DRMS are PBS.

Site		Processor	Nodes	Speed
BIFI	ES	Intel P.IV	56	3.2GHz
CESGA	ES	Intel P.III	16	500MHz
CGG	FR	Intel P.III	58	1.2GHz
CIEMAT	ES	Intel Xeon	226	3.2GHz
GRIF	FR	Intel P.IV	14	2.8GHz
JINR	RU	Intel P.D	30	2.8GHz
L.-HEP	UK	Intel P.IV	374	3GHz
PNPI	RU	Intel P.IV	60	3GHz
RAL	UK	Intel P.IV	62	2.8GHz
RALPP	UK	Intel P.III	1064	1GHz
ScotGRID	UK	Intel Xeon	6	2.8GHz
SINP	RU	Intel Xeon	94	2.8GHz

Table 2. Input file sizes and number of tasks for each database division.

DB Div.	Mean Size	Tasks
10	44MB	45
12	36.5MB	66
14	31.5MB	91
16	27.5MB	120
18	24.5MB	153
20	22MB	190
22	20MB	231

divisions (Table 2). However, the *cd-hit* and *cd-hit-2d* executable file sizes are both 1.1MB.

As can be seen in Figure 1, increasing the number of database divisions favors the parallelism level of the application, although the computation to file transfer ratio worsens. Moreover, the time a task waits in the remote queuing system is not related to the number of divisions as it only depends on the remote resource load status, which is high because the EGEE infrastructure is at production level. Therefore, the queuing time is the most significant part of the walltime, not considering transfer times, of each task in all the experiments.

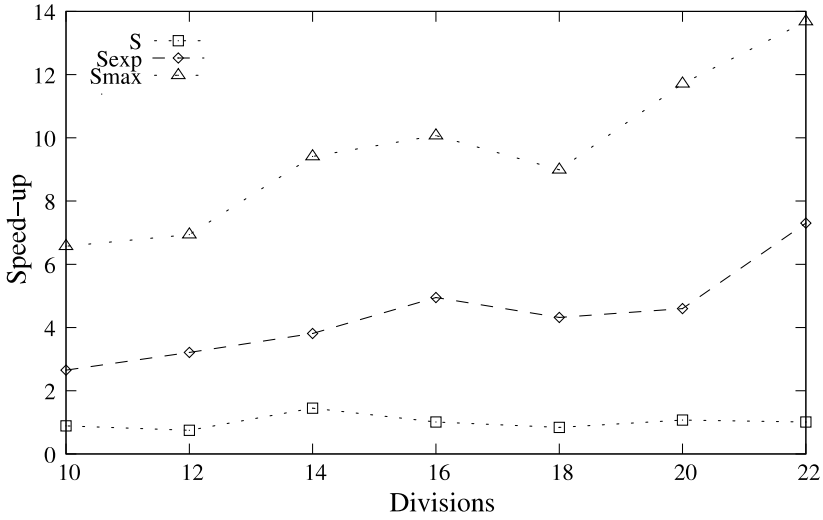


Fig. 2. Speed-up of the workflow execution for different number of database divisions.

Furthermore, we analyzed the potential speed-up that could be obtained for this kind of applications, where the level of parallelism is not constant. To this end, we consider: the speed-up (S) which is the speed gain of the workflow through the Grid approach, compared to the execution on a single machine; the speed-up calculated without considering either queue wait times or job failures (S_{exp}); and an upper bound limit (S_{max}) taking into account just the *critical path*. These three values are represented in Figure 2. The obtained speed-up is very limited, principally due to the file transfer times and the job reschedules. Job reschedules, represented in Figure 3, are due to execution errors (i.e. middleware failures) or suspension timeouts (a job waits in the remote queue more than 5 minutes).

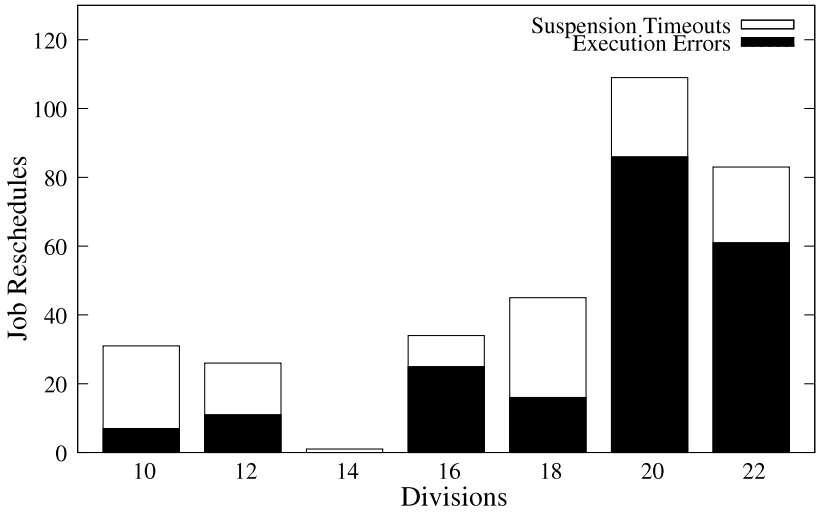


Fig. 3. Number of jobs rescheduled in each experiment.

Summarizing, these results correspond to the analysis of a Bioinformatics workflow porting to a production Grid environment. Due to memory restrictions, this kind of workflow computations cannot be accomplished in a single computer so the Grid approach is still valid, even wih these initial performance results. Anyway, Grid$W$ay is proven to be a robust and reliable workflow engine and its use is encouraged for the next steps that must be performed.

3 Proposed Strategies

The approach with the RefSeq protein database portion resulted in the acquisition of a simple model. Nevertheless, CNIO proposed the analysis of various

meta-genomes, starting with the first published one from Sargasso Sea. This time, the input database contains up to 4,186,284 proteins (1.7GB). Even with a smaller database, the model needs to be revisited and optimized.

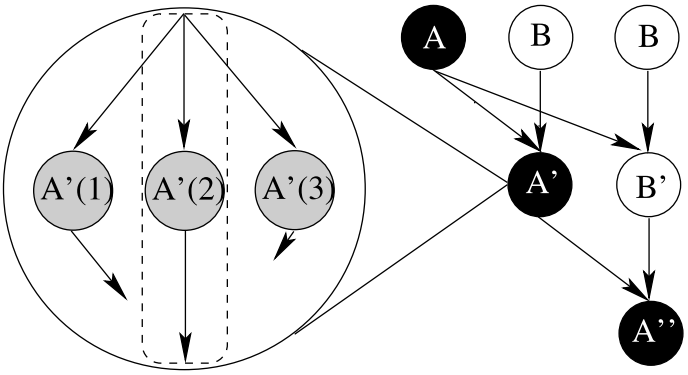


Fig. 4. The *redundancy* strategy.

We divide the optimization strategies in two main types: *redundancy* and *agglomeration*. With the first strategy described in Figure 4, *redundant* tasks are being created for specific nodes of the workflow. When one of these *redundant* task ends, the node is considered executed and the rest of copies are killed. In this way, the more submitted *redundant* tasks, the higher is the possibility for a node to end shortly as not all of its *redundant* tasks would be executed in the same resource. Variants of this strategy include the replication of all the nodes, only those from the *critical path*, and finally, tasks above a defined *blocking* threshold. The *blocking* threshold is the number of tasks that depend on a single one above which, this thask should be replicated. The number of tasks *blocked* by a single node of the workflow is calculated by:

$$b_{i,j} = \begin{cases} ST(N-i) & \text{if } i = j \\ (i-1) + ST(N-i) & \text{if } i \neq j \end{cases} \quad (1)$$

being i the column and j the level where the node is located in the workflow, represented in the example at Figure 5. N is the number of workflow levels. Finally, $ST(n)$ is the *blocking subtree* generated by a node integrating the *critical path* and can be estimated as:

$$ST(n) = \frac{n(1+n)}{2} \quad (2)$$

which is the sum of terms belonging to an arithmetic progression.

Additionaly, we propose the *agglomeration* strategy described in Figure 6, where all the tasks beyond a specific level aren't executed on the Grid but locally.

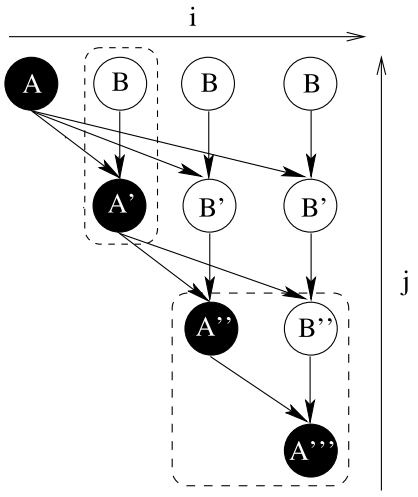


Fig. 5. In this example, B blocks A'. Then A'', A''' and B'' belong to the *blocking subtree* generated by A'.

This decision is taken when the overhead imposed by the Grid is higher than the local execution of the some tasks. This overhead can be estimated by taking the times either from the execution of similar workflows or even from the already finished tasks. Finally, the use of *redundancy* or *agglomeration* shouldn't be exclusive. A combination of both can throw interesting results as the employment of one strategy doesn't restrict the other and may provide cumulative benefits.

4 Conclusions and Future Work

In this contribution we have described a Bioinformatics application performing protein clustering that was successfully ported to the Grid. The nodes of the implemented workflow are distributed and executed among the different Grid resources in order to bypass the memory restrictions inherent to a single machine. Even if times obtained are not low enough compared to local executions, the restrictions mentioned before and the growing input size of the protein databases needed for production purposes, make Grid Computing still stand as a good solution.

At this point, the model, understood as the algorithm's implementation, needs to be optimized. Two strategies with variants were introduced in this contribution. It is our idea to implement all of them so a further study can be performed, not only with every single solution, but also combining them.

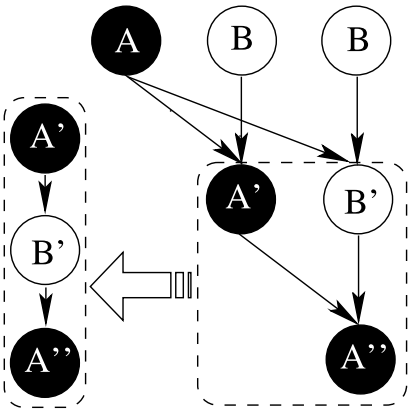


Fig. 6. The *agglomeration* strategy.

References

1. Yu, J., Buyya, R.: A Taxonomy of Workflow Management Systems for Grid Computing. Grid Computing **3** (2005) 171–200
2. Vázquez-Poletti, J.L., Huedo, E., Montero, R.S., Llorente, I.M.: Workflow Management in a Protein Clustering Application. In: Proc. 5th International Workshop on Biomedical Computations on the Grid (BioGrid'07) on the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007), IEEE CS (2007) to appear
3. Li, W., Godzik, A.: CD-HIT: a Fast Program for Clustering and Comparing Large Sets of Protein or Nucleotide Sequences. Bioinformatics **22** (2006) 1658–1659
4. Thain, D., Tannenbaum, T., Livny, M.: Condor and the Grid. In: Grid Computing. John Wiley & Sons, Inc. (2003) 299–335
5. Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Patil, S., Su, M.H., Vahi, K., Livny, M.: Pegasus: Mapping Scientific Workflows onto the Grid. In: Proc. Second European AcrossGrids Conference (AxGrids 2004). Volume 3165 of Lecture Notes in Computer Science. (2004) 11–20
6. Taylor, I., Shields, M., Wang, I.: Resource Management for the Triana Peer-to-Peer Services. In Nabrzyski, J., Schopf, J.M., Węglarz, J., eds.: Grid Resource Management. Kluwer Academic Publishers (2004) 451–462
7. Kurowski, K., Ludwiczak, B., Nabrzyski, J., Oleksiak, A., Pukacki, J.: Dynamic Grid Scheduling with Job Migration and Rescheduling in the GridLab Resource Management System. Scientific Programming (AxGrids 2004 Special Issue) **12** (2004) 263–273
8. McGough, S., Young, L., Afzal, A., Newhouse, S., Darlington, J.: Workflow Enactment in ICENI. In: Proc. UK e-Science All Hands Meeting. (2004) 894–900
9. Laszewski, G.V., Amin, K., Hategan, M., Zaluzec, N., Hampton, S., Rossi, A.: Gridant: A Client-Controllable Grid Workflow System. In: Proc. 37th Annual Hawaii International Conference on System Sciences (HICSS'04). (2004)

10. Yu, J., Buyya, B.: A Novel Architecture for Realizing Grid Workflow using Tuple Spaces. In: Proc. 5th IEEE/ACM International Workshop on Grid Computing (Grid 2004). (2004)
11. Huedo, E., Montero, R.S., Llorente, I.M.: A Framework for Adaptive Execution on Grids. Software – Practice and Experience (SPE) **34** (2004) 631–651
12. Pinedo, M.: Scheduling: Theory, Algorithms, and Systems. Second edn. Prentice Hall, New Jersey, NJ (2002)

A.7. Workflow Management in a Protein Clustering Application

Cita Completa

J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *Workflow Management in a Protein Clustering Application*. 5th International Workshop on Biomedical Computations on the Grid (BioGrid'07) on the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007), Rio de Janeiro (Brasil), Mayo 2007. IEEE Computer Society Press, pp. 679-684.

Abstract

Bioinformatics is demanding more computational resources day after day. The problems proposed by this area are growing in such complexity that traditional computing systems are not able to face them. For solving complex problems which can be divided in tasks with dependencies, a workflow management system must be employed. In this paper, we introduce the use of the workflow management of the GridWay metascheduler for running a Bioinformatics application which implements a complex algorithm performing protein clustering in order to obtain non-redundant protein databases. The use of a general purpose meta-scheduling system will provide the application the fault-tolerance and advance scheduling capabilities needed to execute on a highly dynamic, heterogeneous and faulty environment. The execution results on a production Grid (the EGEE infrastructure) shows the dramatic impact of remote queue waiting times on the application performance; and the critical need of efficient re-scheduling capabilities.

Referencia de Citas Bibliográficas

Deelman et al. (2001); Huedo et al. (2004); Kurowski et al. (2004); Laszewski et al. (2004); Li et al. (2001); McGough et al. (2004); Pinedo (2002); Tayloret al. (2004); Thain et al. (2003); Vázquez-Poletti et al. (2006); Yu y Buyya (2001); Yu y Buyya (2005)

Valoración de la Congreso

Este congreso recibe una valoración de 0.56 por parte del Computer Science Conference Ranking [1]

[1] http://www.cs-conference-ranking.org/

Tasa de Aceptación

Los workshops de este congreso recibieron un total de 91 contribuciones, de los cuales se aceptaron 38 (un 42 %).

Workflow Management in a Protein Clustering Application*

J. L. Vázquez-Poletti E. Huedo R. S. Montero I. M. Llorente

Departamento de Arquitectura de Computadores y Automática
Facultad de Informática, Universidad Complutense de Madrid
28040 Madrid, Spain

Abstract

Bioinformatics is demanding more computational resources day after day. The problems proposed by this area are growing in such complexity that traditional computing systems are not able to face them. For solving complex problems which can be divided in tasks with dependencies, a workflow management system must be employed. In this paper, we introduce the use of the workflow management of the GridWay metascheduler for running a Bioinformatics application which implements a complex algorithm performing protein clustering in order to obtain non-redundant protein databases. The use of a general purpose meta-scheduling system will provide the application the fault-tolerance and advance scheduling capabilities needed to execute on a highly dynamic, heterogeneous and faulty environment. The execution results on a production Grid (the EGEE infrastructure) shows the dramatic impact of remote queue waiting times on the application performance; and the critical need of efficient re-scheduling capabilities.

1 Introduction

As Grid Computing is consolidating and extending its application to many research areas, the complexity of tasks needed to be executed in such environments is growing. The workflow concept appears to satisfy these needs as a way to automate procedures in data transfer and job execution. Then, a workflow management system is the one which not only defines, but manages and executes workflows over a Grid [14].

In a workflow management system, the workflow should be easily designed as a dependency graph, where each node represents one or more computational tasks. Also, it must be capable to provide additional mechanisms for resource information retrieval, data transfer, task scheduling and execution, and fault tolerance. In general, the providing of these mechanisms is challenging because of the nature of the Grid itself, namely: dynamic resource availability and load, heterogeneity and a high fault rate.

The objective of this paper is to show the efficient execution of a Bioinformatics workflow by the GridWay [3]'s workflow management system, which is briefly described in Section 2. This workflow pertains to a toolset called *cd-hit* [6] which is actively used by the Spanish National Oncology Research Center (Centro Nacional de Investigaciones Oncológicas - CNIO) [1]. This toolset is composed by several sub-applications which take a protein sequence database and then eliminate redundant entries. Its algorithm is explained in Section 3.

The growing database size (increasing everyday) makes *cd-hit* infeasible to be executed on a single machine due to its memory requirements and total execution time. Also, the complexity of the algorithm implemented by its workflow is demanding the use of a high number of Grid resources for execution and data transfers. Experimental results of the execution of the *cd-hit* toolset in the EGEE testbed are discussed in Section 4.

Finally, an overview of other workflow management

*This research was supported by Consejería de Educación of Comunidad de Madrid, Fondo Europeo de Desarrollo Regional (FEDER) and Fondo Social Europeo (FSE), through BioGridNet Research Program S-0505/TIC/000101, and by Ministerio de Educación y Ciencia, through research grant TIN2006-02806. Also, this work makes use of results produced by the Enabling Grids for E-sciencE project, a project co-funded by the European Commission (under contract number INFSO-RI-031688) through the Sixth Framework Programme. EGEE brings together 91 partners in 32 countries to provide a seamless Grid infrastructure available to the European research community 24 hours a day. Full information is available at http://www.eu-egee.org/.

[1] http://www.cnio.es/

Seventh IEEE International Symposium on Cluster Computing and the Grid(CCGrid'07)
0-7695-2833-3/07 $20.00 © 2007 IEEE

systems is presented in Section 5, and the conclusions can be found in Section 6.

2 GridWay's Workflow Management

Grid*W*ay is a grid metascheduler which stands on top of Globus services and that has been successfully used, not only in Bioinformatics [2], but also in other research areas [12]. In this section, we describe its workflow management according to the taxonomy proposed in [14]. Grid*W*ay natively handles Directed Acyclic Graph (DAG) based workflows, where each node is a task where its beginning is subject to (depends on) other tasks' finalization. Additionally, Grid*W*ay allows advanced flow structures like loops or branches by using its implementation of the Distributed Resource Management Application API (DRMAA), which is an Open Grid Forum [2] standard. Considering the workflow specification, Grid*W*ay uses an *abstract model* because the workflow is specified without referring to specific Grid resources for task execution. When converting abstract workflows into concrete ones, Grid*W*ay implements a *dynamic scheme* as it uses both dynamic and static information about resources. Scheduling decisions are made at run-time considering the requirements for each task and ranking expresions (a function to determining the resource assignment priority). Inside this scheme, Grid*W*ay employs *just in-time scheduling* which only makes decisions at the time of task execution. Moreover, historical information about task execution is considered. Grid*W*ay's scheduling architecture, as defined in [14], is *centralized* because one central workflow scheduler makes decisions for all tasks.

On the other hand, decision making is considered *local* because only takes each task information into account when scheduling them, as Grid*W*ay resolves task dependencies and starts working at task-level. Once a task (or graph node) is assigned to a resource, the executable and input files are staged onto the remote machine and the execution takes place. When the task finishes, the output files are staged back and Grid*W*ay checks if this event frees the dependencies of other tasks so this process starts again [3]. Grid*W*ay offers automatic staging mechanisms. In particular, a centralized approach is taken as intermediate data is transferred between resources via a checkpoint server to restart the workflow in case of failure.

Fault tolerance in Grid*W*ay must be considered at task-level taking into account possible failures such as network outage or remote and local machine crash. Grid*W*ay applies the following techniques, following the notation of [14]: *retry* (tries the task execution or file transfer on the same resource in case of failure), *alternate resource* (submits a failed task to an alternate resource) and *checkpoint/restart* (failed tasks are moved transparently to other resources).

[2] http://drmaa.org/

3 The Bioinformatics Application

The Bioinformatics application considered in this work performs protein clustering in order to eliminate redundancies in a protein database by comparing its entries [6]. In UniProt[3], which is the world's most comprehensive catalog of information on proteins, *cd-hit* is used to generate the UniRef reference data sets. For parallel processing purposes, a set of separate tools which perform the database division and each singular operation of the algorithm is provided.

The algorithm is represented in Figure 1. First off, a tool called *cd-hit-div* performs the protein database division. The first division, the representative one, is passed to the *cd-hit application* so it's compared to itself (represented as the A task in Figure 1). The output is then compared to the rest of partitions through the *cd-hit-2d* tool (represented as the B tasks). The first partition which results from the last operation is then the representative one (the A' task in Figure 1) and the process starts over until there aren't any more partitions. When the last comparison is performed, all the outputs of the *cd-hit* tool are merged with the *clstr_merge.pl* tool.

With the purpose of porting the application to the Grid, tasks have been divided in two types. In the first type, the job executes both *cd-hit-2d* and *cd-hit* over the given database division. In the second type, the job executes just *cd-hit-2d*. Both the database division and final merging are performed locally. As can be understood from Figure 1, tasks from a certain level cannot start until the first type job from the previous level is not finished, so task dependencies must be managed in this workflow. In a workflow, the node path translated into a sequence of tasks, to which the completion time is subject, is called *critical path* [8]. In this case, the *critical path* is composed by the execution of the first type tasks (the shaded nodes in Figure 1).

4 Experimental Results

The input protein database is a part of the RefSeq database[4] provided by the National Center for

[3] http://www.pir.uniprot.org/database/DBDescription.shtml
[4] http://www.ncbi.nlm.nih.gov/RefSeq/

Seventh IEEE International Symposium on Cluster Computing and the Grid(CCGrid'07)
0-7695-2833-3/07 $20.00 © 2007 IEEE

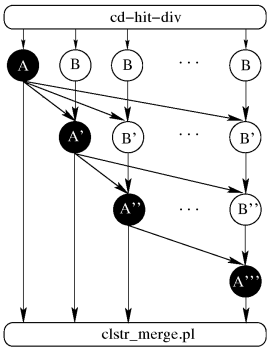


Figure 1. The cd-hit algorithm. White tasks execute *cd-hit-2d* and shaded tasks execute both *cd-hit-2d* and *cd-hit*.

Biotechnology Information (NCBI). Its size is 435MB and stores 504.876 proteins. Moreover, the input file size and the number of the resulting tasks depend on the number of database divisions. Table 1 shows these figures for the experiments performed. For the sake of completeness, the *cd-hit* and *cd-hit-2d* executable file sizes are both 1.1MB.

The experiments where run on resources (see Table 2) from the Enabling Grids for E-sciencE (EGEE) project, as Grid*W*ay can be used in this testbed [11]. The goal of this project is to build the most large production-level grid with great levels of performance and reliability. The tasks were launched, one experiment per division number, from the Universidad Complutense de Madrid at different times on different days of the week during July 2006.

Table 1. Input file sizes and number of tasks for each database division.

DB Div.	Mean Size	Tasks
10	44MB	45
12	36.5MB	66
14	31.5MB	91
16	27.5MB	120
18	24.5MB	153
20	22MB	190
22	20MB	231

Table 2. Testbed resources. All DRMS are PBS.

Site		Processor	Nodes	Speed
BIFI	ES	Intel P.IV	56	3.2GHz
CESGA	ES	Intel P.III	16	500MHz
CGG	FR	Intel P.III	58	1.2GHz
CIEMAT	ES	Intel Xeon	226	3.2GHz
GRIF	FR	Intel P.IV	14	2.8GHz
JINR	RU	Intel P.D	30	2.8GHz
L.-HEP	UK	Intel P.IV	374	3GHz
PNPI	RU	Intel P.IV	60	3GHz
RAL	UK	Intel P.IV	62	2.8GHz
RALPP	UK	Intel P.III	1064	1GHz
ScotGRID	UK	Intel Xeon	6	2.8GHz
SINP	RU	Intel Xeon	94	2.8GHz

Let us first consider the Grid execution of each node of the *cd-hit* workflow. As can be expected, the behavior of the algorithm depends on the number of initial database divisions. Note that, as it increases, the parallelism level of the application is favored; although the computation to file transfer ratio gets worse. This fact is clearly shown in Figure 2, where the average CPU (T_{cpu}), file transfer (T_{xfr}) and queuing (T_{que}) times are presented for different number of database divisions. Moreover, the time a task waits in the remote queuing system is not related to the number of divisions as it only depends on the remote resource load status, which is high because the EGEE infrastructure is at production level. Therefore, the queuing time is the most significant part of the walltime, not considering transfer times, of each task in all the experiments.

In order to analyze the impact of the above considerations in the workflow walltime, let us define the *expected walltime* (T_{exp}) without considering the task queuing times and job failures. This time can be estimated by taking into account that the completion of each level of the workflow is subject to the most significant node's execution:

$$T_{exp} = N \cdot (T^A_{cpu} + T^A_{xfr}), \quad (1)$$

where N is the number of database divisions, and T^A_{cpu} and T^A_{xfr} are the mean CPU and transfer times of the shaded tasks in Figure 1. Also, a lower bound estimation of the walltime is: $T_{min} = N \cdot T^A_{cpu}$. Finally, we can compare them with the sequential execution of the

Seventh IEEE International Symposium on Cluster Computing and the Grid(CCGrid'07)
0-7695-2833-3/07 $20.00 © 2007 IEEE

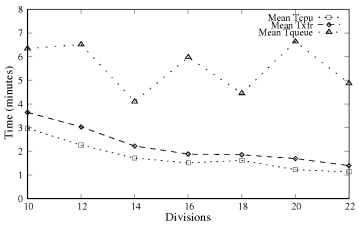


Figure 2. Average CPU (T_{cpu}), file transfer (T_{xfr}) and queuing (T_{que}) times for the workflow tasks and different number of database divisions.

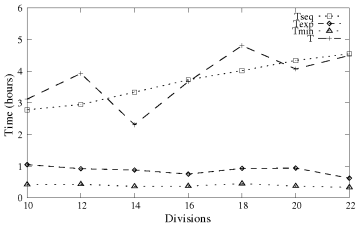


Figure 3. Workflow execution times for different number of database divisions.

workflow (T_{seq}), that can be easily estimated with:

$$T_{seq} = N \cdot T^A_{cpu} + T^B_{cpu} \cdot \sum_{n=2}^{N}(n-1), \qquad (2)$$

where T^B_{cpu} is the CPU time of the white tasks in Figure 1. The CPU times in Equation 2 are measured in the testbed's fastest machine (Intel Pentium IV 3.2GHz).

Figure 3 shows the previous times along with the experimental time (T) obtained in the execution of the workflow for different number of database divisions. As can be observed the Grid execution time of the workflow is similar to that obtained in a single machine; and away from that predicted by equation 1. This is mainly due to the overhead imposed by the remote resource management systems, as shown in Figure 2.

The walltime of the workflow depends also on the

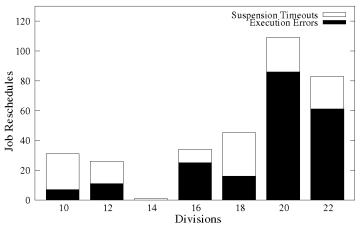


Figure 4. Number of jobs rescheduled in each experiment.

number of times a job is rescheduled to other resource. In our case, these reschedules are due to execution errors (i.e. middleware failures) or suspension timeouts (a job waits in the remote queue more than 5 minutes). The number of jobs rescheduled in each experiment are shown in Figure 4. The influence of these two factors can be clearly observed in Figure 5 for the workflow execution with 14 database divisions.

It is interesting to analyze the potential speed-up that could be obtained for this kind of applications, where the level of parallelism is not constant. Note that the number of tasks decreases in each level of the workflow. To this end, we will consider: the speed-up (S) of the workflow, that obtained without considering queue wait times or job failures (S_{exp}), and an upper bound limit (S_{max}) computed using T_{min}. These three values are represented in Figure 5. As previously discussed, the speed-up obtained by the workflow is very limited. Moreover, the file transfer times also impose a significant reduction of the expected speed-up, when compared to the upper bound limit, S_{max}.

5 Other Workflow Management Systems

Many workload management systems have been developed to execute complex jobs in Grid infrastructures. These projects are generally based on custom middleware developments which made assumptions on the underlying infrastructure. The use of a general-purpose meta-scheduling system have shown its reliability and robustness to execute workflows in a production Grid infrastructures (Globus-based). The application takes advantage of the fault-tolerance, advance scheduling and deployment features of the GridWay meta-scheduler.

Seventh IEEE International Symposium on Cluster Computing and the Grid(CCGrid'07)
0-7695-2833-3/07 $20.00 © 2007 IEEE

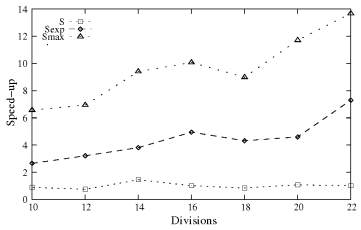


Figure 5. Speed-up of the workflow execution for different number of database divisions.

The Directed Acyclic Graph Manager (DAGMan) [10], provided by Condor, allows users to define jobs with dependencies, being Condor the one who executes each job. DAGMan offers fault tolerance with rescue DAG generation and a definable number of job execution retries.

Pegasus [1] is released as a part of the GriPhyN Virtual Data System (VDS) and extends DAGMan. It generates an executable workflow from the mapping of an abstract workflow to available Grid resources where artificial intelligence planning techniques may be used. In order to find available resources and needed data, Pegasus accesses various Grid information services such as the Globus Monitoring and Discovery Service (MDS) and the Replica Location Service (RLS), as well as the GriPhyN Transformation Catalog and Metadata Catalog Service (MCS). Then, the resource selection is performed both randomly and through a performance prediction infrastructure. Moreover, pluggable task scheduling strategies and just in-time scheduling are supported.

With Triana [9], code can run either locally or distributedly following a parallel or peer-to-peer policy. Tasks are dynamically allocated and both information retrieval and fault tolerance mechanisms are based on the Grid Application Toolkit (GAT) from GridLab [4].

In ICENI [7], several scheduling algorithms are provided such as random, best of n random, simulated annealing and game theory. Moreover, new algorithms can be plugged. Historical data can be used in scheduling as performance is being monitored for each resource and the user may specify metrics. Two scheduling schemes can be found in ICENI: lazy and advanced reservation using WS-Agreement.

GridAnt [5] extends the Ant deploy tool with new components and vocabulary. The information retrieval is performed through Globus MDS and the user defines fault tolerance mechanisms as the architecture of GridAnt is designed for user extension.

The hierarchical scheduling in Gridbus [13] uses the tuple-space model [13]. Gridbus workflow accesses the Grid Market Directory (GMD) in order to retrieve resource information including its access cost. On the other hand, for accounting and billing purposes, the Grid Bank (GB) service can be accessed as well. Failed tasks can be rescheduled to alternative resources. Finally, users can define quality of service constraints such as deadline and cost budget.

6 Conclusions

Grid computing is a matured technology that allows users to run embarrassingly distributed applications. As long as Grid computing reached these first needs, problems that must be faced are gaining more complexity. In this paper we have analyzed the porting of a bioinformatics workflow to a production Grid environment. This kind of workflow computations can not be performed for large protein databases in a single computer due to memory restrictions.

Grid Computing served to process these database divisions separately. However, the efficiency that could be expected is dramatically limited by the nature of the Grid itself: dynamism (queue times), heterogenity and high fault rate. Among the failures we found in our experiments, there were authentication errors, and the loose of callbacks and exit codes. In this study, the GridWay workflow engine has shown to be robust and reliable, as the computation of mid-size databases could be successfully carried out.

Acknowledgements

We would like to end this contribution thanking all the sites belonging to EGEE, in particular those whose machines participated in the experiments.

References

[1] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.-H. Su, K. Vahi, and M. Livny. Pegasus: Mapping Scientific Workflows onto the Grid. In *Proc. Second European AcrossGrids Conference (AxGrids 2004)*, volume 3165 of *Lecture Notes in Computer Science*, pages 11–20, 2004.

[2] E. Huedo, U. Bastolla, R. Montero, and I. Llorente. Computational Proteomics on the Grid. *New Generation Computing, special issue on Grid Systems for Life Sciences*, 22:191–192, 2004.

[3] E. Huedo, R. S. Montero, and I. M. Llorente. A Framework for Adaptive Execution on Grids. *Software – Practice and Experience (SPE)*, 34(7):631–651, 2004.
[4] K. Kurowski, B. Ludwiczak, J. Nabrzyski, A. Oleksiak, and J. Pukacki. Dynamic Grid Scheduling with Job Migration and Rescheduling in the GridLab Resource Management System. *Scientific Programming (AxGrids 2004 Special Issue)*, 12(4):263–273, 2004.
[5] G. V. Laszewski, K. Amin, M. Hategan, N. Zaluzec, S. Hampton, and A. Rossi. Gridant: A Client-Controllable Grid Workflow System. In *Proc. 37th Annual Hawaii International Conference on System Sciences (HICSS'04)*, 2004.
[6] W. Li, L. Jaroszewski, and A. Godzik. Clustering of Highly Homologous Sequences to Reduce the Size of Large Protein Databases. *Bioinformatics*, 17:282–283, 2001.
[7] S. McGough, L. Young, A. Afzal, S. Newhouse, and J. Darlington. Workflow Enactment in ICENI. In *Proc. UK e-Science All Hands Meeting*, pages 894–900, 2004.
[8] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Prentice Hall, New Jersey, NJ, second edition, 2002.
[9] I. Taylor, M. Shields, and I. Wang. Resource Management for the Triana Peer-to-Peer Services. In J. Nabrzyski, J. M. Schopf, and J. Węglarz, editors, *Grid Resource Management*, pages 451–462. Kluwer Academic Publishers, 2004.
[10] D. Thain, T. Tannenbaum, and M. Livny. *Grid Computing*, chapter Condor and the Grid, pages 299–335. John Wiley & Sons, Inc., 2003.
[11] J. L. Vázquez-Poletti, E. Huedo, R. S. Montero, and I. M. Llorente. Coordinated Harnessing of the IRISGrid and EGEE Testbeds with GridWay. *Parallel and Distributed Computing*, 66(5):763–771, 2006.
[12] J. L. Vázquez-Poletti, E. Huedo, R. S. Montero, and I. M. Llorente. Massive Ray Tracing in Fusion Plasmas on EGEE. In *Proc. EGEE (Enabling Grids for E-sciencE) User Forum 2006*, 2006.
[13] J. Yu and B. Buyya. A Novel Architecture for Realizing Grid Workflow using Tuple Spaces. In *Proc. 5th IEEE/ACM International Workshop on Grid Computing (Grid 2004)*, 2004.
[14] J. Yu and R. Buyya. A Taxonomy of Workflow Management Systems for Grid Computing. *Grid Computing*, 3(3–4):171–200, 2005.

Seventh IEEE International Symposium on Cluster Computing and the Grid(CCGrid'07)
0-7695-2833-3/07 $20.00 © 2007 IEEE

A.8. Replication Heuristics for Efficient Workflow Execution on Grids

Cita Completa

J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *Replication Heuristics for Efficient Workflow Execution on Grids*. International Conference on Grid computing, high-performAnce and Distributed Applications at on the Move Federated Conferences (GADA) 2007, Vilamoura (Portugal), Noviembre 2007. Lecture Notes in Computer Science (LNCS). Volume 4805, pp. 31-32, 2007. Springer Verlag.

Abstract

Among the different heuristics available for optimizing workflow execution, the replication ones have been previously used in heterogeneous environments with good results. In this work, we analyze its use for workflow scheduling on Grid infrastructures. In particular, we study its applications to an intree workflow, generated by the distribution of the CD-HIT application. The experiments were conducted on a testbed made of resources from two different grids and results show a significant reduction of the workflow execution time.

Referencia de Citas Bibliográficas

Vázquez-Poletti et al. (2007); Li y Godzik (2006); Huedo et al. (2004); Vázquez-Poletti et al. (2006)

Tasa de Aceptación

Este congreso recibió 55 contribuciones, de los cuales se aceptaron 36 (un 65,45 %).

Replication Heuristics for Efficient Workflow Execution on Grids*

J.L. Vázquez-Poletti, E. Huedo, R.S. Montero, and I.M. Llorente

Departamento de Arquitectura de Computadores y Automática. Facultad de Informática, Universidad Complutense de Madrid. 28040 Madrid, Spain

Abstract. Among the different heuristics available for optimizing workflow execution, the replication ones have been previously used in heterogeneous environments with good results. In this work, we analyze its use for workflow scheduling on Grid infrastructures. In particular, we study its applications to an intree workflow, generated by the distribution of the CD-HIT application. The experiments were conducted on a testbed made of resources from two different grids and results show a significant reduction of the workflow execution time.

In a previous paper [1], we considered a Bioinformatics application, *CD-HIT* (Cluster Database at High Identity with Tolerance) [2], for its porting to the Grid using the Grid*Way* metascheduler [3]. This application performs protein clustering and it can be applied in many activities such as protein family classification, domain analysis, organization of large protein databases or improving database search performance. However, the Grid version of *CD-HIT* didn't provide good performance results, even if it served to bypass memory constrains and so process large data sets. This happened because the nature of the Grid (dynamism, heterogenity and high fault rate).

In this contribution, we apply the *replication* strategy for improving the workflow's efficiency. Supplementary tasks are created for the workflow's *critical path* nodes. When one of these tasks ends, the node is taken as executed and the rest of *replicated* tasks are killed. This way, the more *replicated* tasks are created, the higher is the possibility for that node to be executed shortly by reducing the effect of job failures and queue times.

The input protein database is a compound of UniProt entries and sequence fragments of the Sargasso Sea meta-genome, all of them provided by the National Center for Biotechnology Information (NCBI)[1]. Its size is 1.7GB and it stores

* This research was supported by Consejería de Educación of Comunidad de Madrid, Fondo Europeo de Desarrollo Regional (FEDER) and Fondo Social Europeo (FSE), through BioGridNet Research Program S-0505/TIC/000101, and by Ministerio de Educación y Ciencia, through research grant TIN2006-02806. Also, this work makes use of results produced by the Enabling Grids for E-sciencE project, a project co-funded by the European Commission (under contract number INFSO-RI-031688) through the Sixth Framework Programme. Full information is available at http://www.eu-egee.org/.

[1] http://www.ncbi.nlm.nih.gov/

R. Meersman, Z. Tari, P. Herrero et al. (Eds.): OTM 2007 Ws, Part I, LNCS 4805, pp. 31–32, 2007.

4,186,284 proteins. Focusing on job execution, input file size and job number depend on the number of divisions made to the starting protein database (32, 40 and 48).

For processing the proposed database, two Grid infrastructures were considered: regional and worldwide. Local and regional machines are nearer to the one where the job submission takes place so they offer less latency. On the other hand, machines pertaining to the Enabling Grids for E-siencE (EGEE) infrastructure are more in number and offer more throughput. But, even with busier machines, the EGEE infrastructure guarantees exclusiveness of CPU use. As coordinated harnessing of these infrastructures was retained necessary for the processing of such a big database, the use of Grid*W*ay was still considered, due to its interoperability capabilities [4]. Tasks were launched from Universidad Complutense de Madrid (UCM), belonging to GRIDIMadrid[2], at different times on different days of the week during April 2007. Finally, the maximum number of tasks submitted to a site was limited to 10.

Experimental results show that using the *replication* technique derived in a valuable speed-up. However, this is speed-up was limited by different factors. Firstly and due to scheduling restrictions, the number of simultaneous running jobs was 20. Then, the algorithm's shape made the level of parallelism to decrease. Finally, the Grid's nature itself derived in reschedules due to suspension timeouts and execution errors.

References

1. Vázquez-Poletti, J.L., Huedo, E., Montero, R.S., Llorente, I.M.: Workflow Management in a Protein Clustering Application. In: Proc. 5th Intl. Work. Biomedical Computations on the Grid (BioGrid 2007). 7th IEEE Intl. Symp. Cluster Computing and the Grid (CCGrid 2007), pp. 679–684. IEEE Computer Society Press, Los Alamitos (2007)
2. Li, W., Godzik, A.: CD-HIT: A Fast Program for Clustering and Comparing Large Sets of Protein or Nucleotide Sequences. Bioinformatics 22, 1658–1659 (2006)
3. Huedo, E., Montero, R.S., Llorente, I.M.: A Framework for Adaptive Execution on Grids. Software – Practice and Experience 34, 631–651 (2004)
4. Vázquez-Poletti, J.L., Huedo, E., Montero, R.S., Llorente, I.M.: Coordinated Harnessing of the IRISGrid and EGEE Testbeds with GridWay. J. Parallel and Distributed Computing 66, 763–771 (2006)

[2] http://www.gridimadrid.org/

A.9. CD-HIT Workflow Execution on Grids using Replication Heuristics

Cita Completa

J. L. Vázquez-Poletti, E. Huedo, R. S. Montero and I. M. Llorente. *CD-HIT Workflow Execution on Grids using Replication Heuristics*. 1st International Workshop on Modern Computer Tools for the Biosciences - A Grid Perspective - (ModernBio08) on the 8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2008), Lyon (Francia), Mayo 2008. IEEE Computer Society Press, pp. 735-740.

Abstract

Grid Computing has proven to be a solution for big workflow execution, especially in Bioinformatics. However, Grid nature itself introduces overheads that make its use in many cases an unfeasible solution if considering wall-time. Different heuristics such as list scheduling, agglomeration and replication are available for optimizing workflow execution. In particular, the replication heuristics have been previously used in heterogeneous environments with good results. In this work, we analyze their use for workflow scheduling on Grid infrastructures. In particular, we study its applications to an intree workflow, generated by the distribution of the CD-HIT application. The experiments were conducted on a testbed made of resources from two different grids and results show a significant reduction of the workflow execution time.

Referencia de Citas Bibliográficas

Ahmad y Kwok (1998); Badaj y Agrawal (2004); Chung y Ranka (1992); Deelman et al. (2004); Hagras y Janecek (2004); Hagras y Janecek (2005); Herrera et al (2004); Huedo et al. (2004); Kwok y Ahmad (2000); Li y Godzik (2006); Malcwicz et al. (2007); Olivier et al. (2002); Pinedo (2002); Sclvakumar y Siva (1994); Sih y Lee (1993); Thain et al. (2003); Topcouglu et al. (2002); Vázquez-Poletti et al. (2006); Vázquez-Poletti et al. (2007); Yu y Buyya (2005)

Valoración de la Congreso

Este congreso recibe una valoración de 0.56 por parte del Computer Science Conference Ranking [2].

[2] http://www.cs-conference-ranking.org/

Tasa de Aceptación

Las contribuciones enviadas a este congreso tuvieron una tasa de aceptación del 57 %.

CD-HIT Workflow Execution on Grids using Replication Heuristics*

J. L. Vázquez-Poletti E. Huedo R. S. Montero I. M. Llorente

Departamento de Arquitectura de Computadores y Automática
Facultad de Informática, Universidad Complutense de Madrid
28040 Madrid, Spain

Abstract

Grid Computing has proven to be a solution for big workflow execution, especially in Bioinformatics. However, Grid nature itself introduces overheads that make its use in many cases an unfeasible solution if considering wall-time. Different heuristics such as list scheduling, agglomeration and replication are available for optimizing workflow execution. In particular, the replication heuristics have been previously used in heterogeneous environments with good results. In this work, we analyze their use for workflow scheduling on Grid infrastructures. In particular, we study its applications to an intree workflow, generated by the distribution of the CD-HIT application. The experiments were conducted on a testbed made of resources from two different grids and results show a significant reduction of the workflow execution time.

1 Introduction

Workflow management systems and Grid Computing are providing solutions to problems proposed by Bioinformatics. Workflow management systems [22] allow the execution of complex applications than can be divided in tasks with data dependencies. Grid Computing, on the other hand, offers the applications access to a great amount of computing resources.

*This research was supported by Consejería de Educación of Comunidad de Madrid, Fondo Europeo de Desarrollo Regional (FEDER) and Fondo Social Europeo (FSE), through BioGridNet Research Program S-0505/TIC/000101, and by Ministerio de Educación y Ciencia, through research grant TIN2006-02806. Also, this work makes use of results produced by the Enabling Grids for E-sciencE project, a project co-funded by the European Commission (under contract number INFSO-RI-031688) through the Sixth Framework Programme. EGEE brings together 91 partners in 32 countries to provide a seamless Grid infrastructure available to the European research community 24 hours a day. Full information is available at http://www.eu-egee.org/.

In a previous paper [21], we considered a Bioinformatics application, *CD-HIT* (Cluster Database at High Identity with Tolerance) [11], for its porting to the Grid. This application performs protein clustering, which consists in removing redundant sequences from a protein database in order to generate a database of only the representatives. Protein clustering can be applied in many activities such as protein family classification, domain analysis, organization of large protein databases or improving database search performance. However, the Grid version of *CD-HIT* didn't provide good performance results, even if it served to bypass memory constraints and so process large data sets. This happened because the nature of the Grid (dynamism, heterogenity and high fault rate).

As optimization is needed in this workflow, described with the previous work in Section 2, we considered well known heuristics that proved to throw good results in other heterogeneous computational infrastructures. These optimization strategies are described in Section 3. However, in Section 4 we focused in the *replication* strategy for optimizing the cited workflow and then, evaluated it through experimental results in Section 5. Finally, some conclusions and future work are shown at the end of the paper.

2 The Application

The *CD-HIT* application was successfully ported to the Grid [21] using the Grid*W*ay metascheduler [9]. However, workflow management systems such as the Directed Acyclic Graph Manager (DAGMan) [18] provided by Condor, and Pegasus [4] were considered among others. In the past, the Grid*W*ay metascheduler has been previously used with good results in many research areas, including Bioinformatics. It natively handles DAG based workflows and allows advanced flow structures like loops or branches. Grid*W*ay offers an implementation of both C and JAVA bindings

of the Distributed Resource Management Application API (DRMAA), which is an Open Grid Forum[1] standard [8].

Regarding the workflow management performed by Grid*Way*, workflow specification follows an *abstract model*, and its concretion, a *dynamic scheme* [22]. Scheduling decisions are taken considering the requirements for each task and resource ranking expressions at run-time. Additionally, historical information about task execution is considered. Grid*Way* provides fault tolerance mechanisms which include trying the task execution or file transfer on the same resource in case of failure, and submitting of a failed task to an alternate resource.

The first approach to the *gridification* of this algorithm, described in a previous publication [21], was a simple case of *list scheduling* where the workflow nodes were given a certain priority and then handled to Grid*Way*. Even if complex techniques may be used for determining task priority [13], in our case, nodes from the *critical path* [15] deserve the highest priority. This priority is given by the order in which jobs are submitted to Grid*Way* at the beginning of the process. Due to dependencies among workflow nodes, not all tasks are submitted at the first moment to remote resources, as they are put *on hold*. Summarizing, priorities are established both by the order jobs are sent to schedule and by their dependencies.

Distributing the workflow nodes over the Grid has proven to bypass the memory limitations intrinsic to a single machine. Nevertheless, execution times measured on the Grid were higher than those obtained locally as Grid environment is highly dynamic, heterogeneous and faulty. In addition, the time a task waits in the remote queuing system can be the most significant part of each task's walltime, specially in infrastructures at production level. Hence the remote queue waiting time completely determines the overall efficiency obtained by the application.

The first experiments were performed with a mid-sized protein database (504,876 proteins, 435MB). However, the Spanish National Oncology Research Center (Centro Nacional de Investigaciones Oncológicas - CNIO)[2] requires the analysis of larger databases. For this purpose, the Grid approach is still valid due to the single machine restrictions mentioned above, but the model had to be revisited considering the optimization heuristics cited at the following Section.

[1] http://www.drmaa.org/
[2] http://www.cnio.es/

3 Workflow Optimization Heuristics

Among the optimization strategies that may apply to a workflow, there are different approaches that can be considered, taking into account the nature of the scheduling problem. In the first approach, called *list scheduling*, priorities are assigned to jobs either statically or dynamically [16]. In general, tasks are not scheduled regarding ulterior ones so this technique doesn't always provide an optimized solution. Algorithms pertaining to this approach are:

- Heterogeneous Earliest Finish Time (HEFT) [19], which schedules tasks minimizing their finishing time in an insertion based manner;
- Critical Path on a Processor (CPOP) [19], that detaches a machine just for critical path tasks;
- Bubble Scheduling and Allocation (BSA) [10], which firstly serializes the task graph and then inserts all the tasks to a processor;
- Dynamic Level Scheduling (DLS) [17], that delays scheduling when the given task is ready;
- Critical Nodes Parent Trees (CNPT) [5], which considers the task earliest execution time;
- Iso-Level Heterogeneous Allocation (ILHA) [14], that allocates to each processor a number of tasks proportional to its computing power.

The second approach is the *agglomeration* technique, which consists in clustering jobs in groups so communication overheads are reduced. Even if this technique is not complex in its implementation, the obtained performance by itself, is not as desiderable as expected [2].

The last approach is the *replication* strategy, that this contribution will focus on. In this technique, some jobs are replicated in order to increase the possibility of speed-up the execution due to a better resource selection. Algorithms inside this strategy are:

- Heterogeneous Critical Tasks Reverse Duplicator (HCTRD) [6], that replicates the parent-tree or some selected parents of the selected task;
- Bottom-up Top-down Duplication Heuristic (BTDH) [3], which assumes that task starting time may be reduced eventually by the replication of all the necessary task ancestors;
- Heterogeneous Critical Parents with Fast Duplicator (HCPFD) [7], that performs the replication considering the idle time left by the selected task on a given machine;

- Critical Path based Full Duplication Algorithm (CPFD) [1], which replicates all possible parents of the considered task.

A version of this last heuristic was chosen because its structure fits the studied Bioinformatics application, and its adoption is presented in the following section.

4 Applying the Replication Heuristic

In this contribution, we apply the *replication* strategy for improving the workflow's efficiency as shown at Figure 1. Again, in this technique supplementary tasks are created for given nodes of the workflow. When one of these tasks ends, the node is taken as executed and the rest of *replicated* tasks are killed. This way, the more *replicated* tasks are created, the higher is the possibility for that node to be executed shortly by reducing the effect of job failures and queue times.

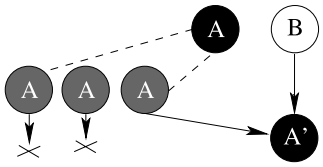


Figure 1. The *replication* technique applied to the CD-HIT algorithm.

Different variants of this strategy can be devised depending on the nodes where *replication* is applied. The most simple variant consists in the replication of all the tasks. In other variant, just the nodes from the *critical path* are replicated. In the last variant, those target nodes above a given *blocking* threshold are replicated. We define the *blocking* value as the number of nodes of the workflow which depend on the execution of that node and it is calculated by:

$$b_{i,j} = \begin{cases} ST(N-i) & \text{if} \quad i = j \\ (i-1) + ST(N-i) & \text{if} \quad i \neq j \end{cases} \quad (1)$$

being i the column and j the level where the node is located in the workflow. N is the number of workflow levels. Finally, $ST(n)$ is the *blocking subtree* generated by a node integrating the *critical path* and can be estimated as:

$$ST(n) = \frac{n(1+n)}{2} \quad (2)$$

which is the sum of terms belonging to an arithmetic progression. An example of this is shown at Figure 1. In the present contribution, the *critical path* variant was employed (Critical Path based Full Duplication Algorithm [1]), creating 3 copies per task.

However, some studies conducted about job *replication* at the same resource state that local *backfilling* mechanisms generate mostly drawbacks [12]. In our case, not all replicated tasks are necessarily sent to the same cluster, so these mechanisms may not always affect global performance. Moreover, when a resource fails, Grid*W*ay implements an exponential linear backoff strategy at resource level, henceforth resources with persistent failures are discarded. The *replication* technique, in conjunction with this scheduling policy, allows a fast *functional* resource discovery at the beginning of the workflow execution.

5 Experimental Results

The input protein database is a compound of UniProt[3] entries and sequence fragments of the Sargasso Sea meta-genome[4], all of them provided by the National Center for Biotechnology Information (NCBI)[5]. Its size is 1.7GB and it stores 4,186,284 proteins. Focusing on job execution, input file size and job number depend on the number of divisions made to the starting protein database. For this contribution, we considered 32, 40 and 48 divisions.

For processing the proposed database, two Grid infrastructures were considered: regional and worldwide, both of them detailed at Table 1. Local and regional machines are nearer to the one where the job submission takes place so they offer less latency. On the other hand, machines pertaining to the Enabling Grids for E-siencE (EGEE)[6] infrastructure are more in number and offer more throughput. But, even with busier machines, the EGEE infrastructure guarantees exclusiveness of CPU use. As coordinated harnessing of these infrastructures was retained necessary for the processing of such a big database, the use of Grid*W*ay was still considered, due to its interoperability capabilities [20]. Experiments were conducted for each database division taken into account and results were compared depending whether the *replication* technique was used or not. Tasks were launched from Universidad Complutense de Madrid (UCM), belonging to GRIDIMadrid[7], at different times on different days of the week during April

[3] http://www.ncbi.nlm.nih.gov/RefSeq/
[4] ftp://ftp.ncbi.nih.gov/genbank/wgs/
[5] http://www.ncbi.nlm.nih.gov/
[6] http://www.eu-egee.org/
[7] http://www.gridimadrid.org/

Table 1. Grid resources from the joint infrastructure employed during the experiment.

Site	Count.	Proc.	Speed	Nodes
GRIDIMadrid Resources				
UCM	ES	P4	3216	2
CIEMAT	ES	P4	2392	22
EGEE Resources (BIOMED Virtual Organization)				
BHAM-UNI	UK	PIII	800	128
BRUNEL	UK	P4	2000	5
CGG	FR	PIII	1266	56
CIEMAT	ES	PIII	1001	220
CYF-KR	PL	P4	2800	264
GRID-ACAD	BG	P4	2400	78
HELLASGRID	GR	P4	3400	356
IFCA	ES	P4	3200	96
II	MK	P4	3300	8
IMPERIAL	UK	P4	2000	188
IN2P3	FR	PIII	1001	569
INFN	IT	P4	2400	124
IPP-ACAD	BG	P4	2800	10
JET-EFDA	UK	PIII	1098	66
KELDYSH	RU	P4	3000	14
L.-HEP	UK	P4	3000	380
LIP	PT	P4	2200	52
MAN-UNI	UK	P4	2800	844
PNPI	RU	P4	3000	112
SAVBA	SK	P4	3200	41
SRCE	HR	P4	2193	16
UAM	ES	P4	2566	14
UCL	UK	P4	2800	312
UNI-LINZ	AT	P4	3014	8

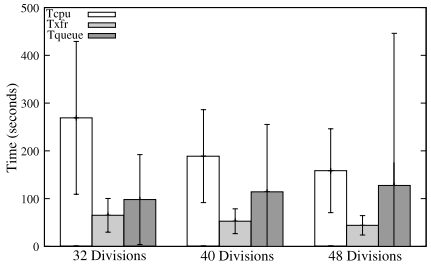


Figure 2. Consolidated CPU (T_{cpu}), file transfer (T_{xfr}) and queuing (T_{queue}) times for the workflow tasks and different database partition number, with and without optimization.

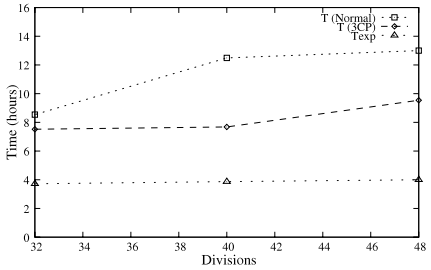


Figure 3. Workflow execution times for different number of database divisions without optimization and using the *replication* strategy (3 copies of each *critical path* task).

2007. Finally, a constrain was added to scheduling: the maximum number of tasks submitted to a site was limited to 10.

Consolidated average CPU (T_{cpu}), file transfer (T_{xfr}) and queuing (T_{queue}) times and their standard deviations for the workflow tasks are shown in Figure 2, categorized by the different number of database partitions. Both T_{cpu} and T_{queue} present a high variability. In the case of T_{cpu}, resources with different computing capacity were available during each experiment. On the other hand, the variability of Local Resource Management Systems (LRMS) located at the resources, affected T_{queue}. T_{xfr} also presents some variability, this is due to the different network links involved.

Figure 3 shows values of different execution times: T is the experimental workflow execution time and T_{exp} is the *expected walltime*, which is explained next. The *expected walltime* (T_{exp}) does take into account only the *critical path* tasks and not considering job failures. The value of T_{exp} is the same for both cases (with and without optimization) and it is calculated as:

$$T_{exp} = N \cdot (T_{cpu} + T_{xfr} + T_{queue}), \tag{3}$$

where N is the number of database divisions. Figure 3 shows that the obtained experimental times applying the optimization are lower than without.

Speed-up can be computed dividing the overall execution time by the workflow's *sequential time* (T_{seq}). It is possible to estimate T_{seq} with:

$$T_{seq} = N \cdot T^A_{cpu} + T^B_{cpu} \cdot \frac{N \cdot (N+1)}{2}, \tag{4}$$

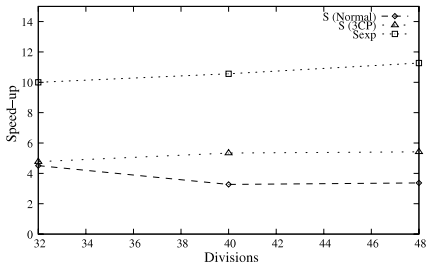


Figure 4. Speed-up of the workflow execution for different number of database divisions, with and without optimization.

where T^A_{cpu} and T^B_{cpu} are the CPU time of shaded and non-shaded tasks as shown in Figure 1. Their values correspond to the average execution times obtained during the experiments. In any case, we would like to remark that sequential processing is not possible to be done on a single CPU with the original *CD-HIT* application due to memory constraints.

Different speed-up estimations are shown at Figure 4: the *experimental speed-up* (S) and that obtained with the *expected walltime* (S_{exp}). Basically, we observe that the optimization allows to raise the level of parallelism. This doesn't happen in the non-optimized case, where S decreases with the number of database divisions. Some aspects must be considered in this study. Firstly, the maximum number of simultaneously used processors was limited to 20 due to scheduling constraints. Then, the level of parallelism of the application decreases on each level because of its in-tree shape, acquiring S_{exp} a maximum value of 12.20 with 48 database divisions.

Despite the factors explained above, the optimization represented about 50% of speed-up gain. The 20 speed-up value couldn't be reached because of the Grid's inherent nature, represented by the number of reschedules (shown at Table 2). On the other hand, the improvement was also reflected on the level completion mean times, shown at the same Table.

Even if replication heuristics provide efficiency to this workflow execution, its cost should be taken into account. Depending on the number of database divisions, we may find different average transfer and execution times associated to the two *discarded* tasks (*replicated* tasks which are killed or do finish before the framework decides to kill them). For 32 divisions, *discarded* tasks consumed an average of 45 seconds of T_{xfr} and 5 minutes 16 seconds of T_{cpu}. For 40 divisions, 30 seconds of T_{xfr} and 3 minutes 51 seconds of T_{cpu}. Finally, for 48 divisions, they consumed an average of 25 seconds of T_{xfr}, and 2 minutes 55 seconds of T_{cpu}.

Table 2. Number of jobs rescheduled in each experiment with optimization (3CP) and without (Normal) and mean times (considering T_{cpu}, T_{xfr}, T_{queue} and task reschedules) for each algorithm level completion.

Divisions	Reschedules		Mean Times	
	Norm.	3CP	Norm.	3CP
32	69	34	16.3'	14.7'
40	170	35	14.6'	11.7'
48	68	27	14.8'	12.1'

Consequently, we may define the cost of replication as the sum of the times mentioned before. The cost for 32 divisions was 8 hours 6 minutes. For 40 divisions, 7 hours 25 minutes. Finally, for 48 divisions, it was 7 hours 55 minutes. We may consider however the worst case for *discarded* tasks, which is to execute them on the slowest computing resource, employing the lowest bandwidth for data transfers. From here, an upper-bound for replication cost can be estimated. For 32 divisions, this value is 32 hours. For 40 divisions, 28 hours. At the end, for 48 divisions, it is 49 hours.

In any case, we would like to note that the additional computational effort mentioned before was performed by spare resources. Therefore, without harming the performance of other Grid applications.

6 Conclusions and Future Work

Grid's inherent nature derives in high queuing times and fault rate. This makes the Grid to be an unfeasible solution for many workflows execution. In this contribution we have reviewed three approaches to optimization techniques. Then, we focused on one of them for a specific type of workflow in order to minimize the effect of the issues addressed before.

Experimental results show that using the *replication* technique derived in a valuable speed-up. However, this speed-up was limited by different factors. Firstly and due to scheduling restrictions, the number of simultaneous running jobs was 20. Then, the algorithm's shape made the level of parallelism decrease. Finally, the Grid's nature itself derived in reschedules due to suspension timeouts and execution errors.

Studying the experimental impact of this optimization strategy is only the beginning. We intend to compare the *replication* variants and then, apply the *agglomeration* strategy. With these steps it is our idea to perform a further evaluation of optimization techniques that could be applied to this particular workflow.

7 Acknowledgments

The authors would like to thank all the institutions involved in the EGEE project and in the GRIDIMadrid infrastructure, in particular those who collaborated in the experiments.

References

[1] I. Ahmad and Y.-K. Kwok. On Exploiting Task Duplication in Parallel Program Scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 9(8):872–892, 1998.

[2] R. Bajaj and D. Agrawal. Improving Scheduling of Tasks in A Heterogeneous Environment. *IEEE Transactions on Parallel and Distributed Systems*, 15(2):107–118, 2004.

[3] Y. Chung and S. Ranka. Application and Performance Analysis of a Compile-Time Optimization Approach for List Scheduling Algorithms on Distributed-Memory Multiprocessors. In *Proc. Supercomputing '92*, pages 512–521. IEEE CS, 1992.

[4] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.-H. Su, K. Vahi, and M. Livny. Pegasus: Mapping Scientific Workflows onto the Grid. In *Proc. 2nd European AcrossGrids Conference (AxGrids 2004)*, volume 3165 of *Lecture Notes in Computer Science*, pages 11–20, 2004.

[5] T. Hagras and J. Janecek. A High Performance, Low Complexity Algorithm for Compile-Time Job Scheduling in Homogeneous Computing Environments. In *Proc. Intl. Conf. on Parallel Processing Workshops (ICPPW'03)*, pages 149–155. IEEE CS, 2003.

[6] T. Hagras and J. Janecek. A Near Lower-Bound Complexity Algorithm for Compile-Time Task-Scheduling in Heterogeneous Computing Systems. In *Proc. 3rd Intl. Symp. Parallel and Distributed Computing, 3rd Intl. Work. Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks*, pages 106–113. IEEE CS, 2004.

[7] T. Hagras and J. Janecek. A High Performance, Low Complexity Algorithm for Compile-Time Task Scheduling in Heterogeneous Systems. *J. Parallel and Distributed Computing*, 65(4):479–491, 2005.

[8] J. Herrera, E. Huedo, R. S. Montero, and I. M. Llorente. Developing Grid-Aware Applications with DRMAA on Globus-based Grids. In *Proc. 10th Intl. Conf. Parallel Processing (Euro-Par 2004)*, volume 3149 of *Lecture Notes in Computer Science*, pages 429–435, 2004.

[9] E. Huedo, R. S. Montero, and I. M. Llorente. A Framework for Adaptive Execution on Grids. *Software – Practice and Experience*, 34(7):631–651, 2004.

[10] Y.-K. Kwok and I. Ahmad. Link Contention-Constrained Scheduling and Mapping of Tasks and Messages to A Network of Heterogeneous Processors. *J. Cluster Computing*, 3(2):113–124, 2000.

[11] W. Li and A. Godzik. CD-HIT: A Fast Program for Clustering and Comparing Large Sets of Protein or Nucleotide Sequences. *Bioinformatics*, 22:1658–1659, 2006.

[12] G. Malewicz, I. Foster, A. L. Rosemberg, and M. Wilde. Benefits and Drawbacks of Redundant Batch Requests. *J. Grid Computing*, 5(2):197–212, 2007.

[13] G. Malewicz, I. Foster, A. L. Rosemberg, and M. Wilde. A Tool for Prioritizing DAGMan Jobs and Its Evaluation. *J. Grid Computing*, 5(2):197–212, 2007.

[14] B. Olivier, B. Vincent, and R. Yves. The Iso-Level Scheduling Heuristic for Heterogeneous Processors. In *Proc. 10th Euromicro Conf. Parallel, Distributed and Network-based Processing (PDP 2002)*, pages 335–350. IEEE CS, 2002.

[15] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Prentice Hall, second edition, 2002.

[16] S. Selvakumar and C. Siva Ram Murthy. Scheduling Precedence Constrained Task Graphs with Non-Negligible Intertask Communication onto Multiprocessors. *IEEE Transactions on Parallel and Distributed Systems*, 5(4):328–336, 1994.

[17] G. Sih and E. Lee. A Compiler-Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures. *IEEE Transactions on Parallel and Distributed Systems*, 4(2):175–186, 1993.

[18] D. Thain, T. Tannenbaum, and M. Livny. *J. Grid Computing*, chapter Condor and the Grid, pages 299–335. 2003.

[19] H. Topcuoglu, S. Hariri, and M.-Y. Wu. Performance-Effective and Low-complexity Task Scheduling for Heterogeneous Computing. *IEEE Transactions on Parallel and Distributed Systems*, 13(3):260–274, 2002.

[20] J. L. Vázquez-Poletti, E. Huedo, R. S. Montero, and I. M. Llorente. Coordinated Harnessing of the IRISGrid and EGEE Testbeds with GridWay. *J. Parallel and Distributed Computing*, 66(5):763–771, 2006.

[21] J. L. Vázquez-Poletti, E. Huedo, R. S. Montero, and I. M. Llorente. Workflow Management in a Protein Clustering Application. In *Proc. 5th Intl. Work. Biomedical Computations on the Grid (BioGrid'07). 7th IEEE Intl. Symp. Cluster Computing and the Grid (CCGrid 2007)*, pages 679–684. IEEE CS, 2007.

[22] J. Yu and R. Buyya. A Taxonomy of Workflow Management Systems for Grid Computing. *J. Grid Computing*, 3(3–4):171–200, 2005.

Bibliografía

[AAF+01] Gabrielle Allen, David Angulo, Ian Foster, Gerd Lanfermann, Chuang Liu, Thomas Radke, Ed Seidel, and John Shalf. The Cactus Worm: Experiments with Dynamic Resource Discovery and Allocation in a Grid Environment. *International Journal of High Performance Computing Applications*, 15(4):345–358, 2001.

[ABCea04] G. Avellino, S. Beco, B. Cantalupo, and et al. The DataGrid Workload Management System: Challenges and Results. *Journal of Grid Computing*, 2(4):353–367, 2004.

[ACea05] R. Alfieri, R. Cecchini, and et al. From gridmap-file to VOMS: Managing Authorization in a Grid Environment. *Future Generation Computing Systems*, 21(4):549–563, 2005.

[AK98] I. Ahmad and Y.-K. Kwok. On Exploiting Task Duplication in Parallel Program Scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 9(8):872–892, 1998.

[And04] S. Andreozzi. GLUE Schema Implementation for the LDAP Data Model. Technical Report INFN/TC-04/16, Instituto Nazionale di Fisica Nucleare (INFN), Sep 2004. Disponible en http://www.lnf.infn.it/sis/preprint/pdf/INFN-TC-04-16.pdf.

[BA04] R. Bajaj and D.P. Agrawal. Improving Scheduling of Tasks in A Heterogeneous Environment. *IEEE Transactions on Parallel and Distributed Systems*, 15(2):107–118, 2004.

[BCD+08] Stephen Burke, Simone Campana, Antonio Delgado, Flavia Donno, Patricia Méndez, Roberto Santinelli, and Andrea Sciabà. gLite 3.1 User Guide. https://edms.cern.ch/document/722398/, 2008.

[BL] I.C. Baianu and H.C. Lin. Computer Simulation and Computability of Biological Systems. *Mathematical Modelling in Biology and Medicine*.

[BWC+03] Francine Berman, Richard Wolski, Henri Casanova, Walfredo Cirne, Holly Dail, Marcio Faerman, Silvia Figueira, Jim Hayes, Graziano Obertelli, Jennifer Schopf, Gary Shao, Shava Smallen, Neil Spring, Alan Su, and Dmitrii Zagorodnov. Adaptive Computing on the Grid Using AppLeS. *IEEE Transactions Parallel Distributed Systems*, 14(4):369–382, 2003.

[Cas05] Francisco Castejon. Massive Ray Tracing in Fusion Plasmas: Memorandum of Understanding. Technical report, CIEMAT, Spain, November 2005.

[CBL+05] C. Cervato, G. Bohling, C. Loepp, T. Taylor, W.S. Snyder, P. Diver, J. Reed, D. Fils, D. Greer, and Xiaoyun Tang. The chronos system: geoinformatics for sedimentary geology and paleobiology. *Local to Global Data Interoperability – Challenges and Technologies*, 0:182–186, 2005.

[CG84] G.M. Church and W. Gilbert. Genomic Sequencing. *Proc. of National Academy of Sciences of the United States of America*, 81(7):1991–1995, 1984.

[CGE+05] Clovis Chapman, Charaka Goonatilake, Wolfgang Emmerich, Matthew Farrellee, Todd Tannenbaum, Miron Livny, Mark Calleja, and Martin Dove. Condor Birdbath - Web Service Interface to Condor. In *Proc. UK E-Science All Hands Meeting*, 2005.

[CGVC04] Stephen R. Comeau, David W. Gatchell, Sandor Vajda, and Carlos J. Camacho. ClusPro: a Fully Automated Algorithm for Protein–Protein Docking. *Nucleic Acids Research*, 32:W96, 2004.

[CLS04] Simone Campana, Maarten Litmaath, and Andrea Sciaba. LCG-2 Middleware Overview. Available at https://edms.cern.ch/document/498079/0.1, 2004.

[CR92] Y.C. Chung and S. Ranka. Application and Performance Analysis of a Compile-Time Optimization Approach for List Scheduling Algorithms on Distributed-Memory Multiprocessors. In *Proc. Supercomputing '92*, pages 512–521. IEEE CS, 1992.

[CTea04] Francisco Castejon, Maxim A. Tereshcenko, and et al. Electron Bernstein Wave Heating Calculations for TJ-II Plasmas. *American Nuclear Society*, 46(2):327–334, 2004.

[DMD+04] Antonio Delgado, Patricia Méndez, Flavia Donno, Andrea Sciabà, Simone Campana, and Roberto Santinelli. LCG-2 User Guide. Available at https://edms.cern.ch/file/454439/2/, 2004.

[Ein61] A. Einstein. *Relativity: The Special and the General Theory*. Three Rivers Press, 1961.

[Elg96] S.C. Elgin. Heterochromatin and Gene Regulation in Drosophila. *Current Opinion in Genetics & Development*, 6:193, 1996.

[ELvD+96] D. H. J. Epema, M. Livny, R. van Dantzig, X. Evers, and J. Pruyne. A Worldwide Flock of condors: Load Sharing among Workstation Clusters. *Future Generation Computer Systems*, 12(1):53–65, 1996.

[FBS04] Xiang Fu, Tevfik Bultan, and Jianwen Su. Analysis of Interacting BPEL Web Services. In *Proc. 13th Intl. Conf. on World Wide Web (WWW '04)*, pages 621–630, 2004.

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. RFC2616: Hypertext Transfer Protocol – HTTP/1.1. Technical report, IETF, 1999. Disponible en http://www.ietf.org/rfc/rfc2616.txt.

[FKT01] Ian Foster, Carl Kesselman, and Steven Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. In *Proc. 7th Int. Conf. on Parallel Processing (Euro-Par 2001)*, pages 1–4. Lecture Notes in Computer Science, 2001.

[FM67] Walter M. Fitch and Emanuel Margoliash. Construction of Phylogenetic Trees. *Science*, 155(3760):279–284, 1967.

[FTL+02] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke. Condor-G: A Computation Management Agent for Multi-Institutional Grids. *Cluster Computing*, 5:237–246, 2002.

[Gee03] D. Geer. Grid Computing using the Sun Grid Engine. Technical report, Technical Enterprises, Inc., 2003.

[GJK+06] Tom Goodale, Shantenu Jha, Harmut Kaiser, Thilo Kielmann, Pascal Kleijer, Gregor von Laszewski, Craig Lee, Andre Merzky, Hrabri Rajic,

and John Shalf. SAGA: A Simple API for Grid Applications, High-Level Application Programming on the Grid. *Computational Methods in Science and Technology*, 12(1):7–20, 2006.

[HBML04] E. Huedo, U. Bastolla, R.S. Montero, and I.M. Llorente. Computational Proteomics on the Grid. *New Generation Computing, special issue on Grid Systems for Life Sciences*, 22:191–192, 2004.

[HBWI07] Matthew R. Helmus, Thomas J. Bland, Christopher K. Williams, and Anthony R. Ives. Phylogenetic Measures of Biodiversity. *American Naturalist*, 169(3):68–83, 2007.

[HJ03] T. Hagras and J. Janecek. A High Performance, Low Complexity Algorithm for Compile-Time Job Scheduling in Homogeneous Computing Environments. In *Proc. Intl. Conf. on Parallel Processing Workshops (ICPPW'03)*, pages 149–155. IEEE CS, 2003.

[HJ04] T. Hagras and J. Janecek. A Near Lower-Bound Complexity Algorithm for Compile-Time Task-Scheduling in Heterogeneous Computing Systems. In *Proc. 3rd Intl. Symp. Parallel and Distributed Computing, 3rd Intl. Work. Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks*, pages 106–113. IEEE CS, 2004.

[HJ05] T. Hagras and J. Janecek. A High Performance, Low Complexity Algorithm for Compile-Time Task Scheduling in Heterogeneous Systems. *J. Parallel and Distributed Computing*, 65(4):479–491, 2005.

[HMHL04] J. Herrera, R.S. Montero, E. Huedo, and I.M. Llorente. DRMAA Implementation within the GridWay Framework. In *Workshop on Grid Application Programming Interfaces, 12th Global Grid Forum (GGF12)*, 2004.

[HML04] E. Huedo, R. S. Montero, and I. M. Llorente. A Framework for Adaptive Execution on Grids. *Software – Practice and Experience*, 34(7):631–651, 2004.

[HML05] Eduardo Huedo, Ruben S. Montero, and Ignacio M. Llorente. An Evaluation Methodology for Computational Grids. In *Proc. 2005 International Conference on High Performance Computing and Communications (HPCC05)*, volume 3726 of *Lecture Notes in Computer Science*, pages 499–504, October 2005.

[HML06a] E. Huedo, R. S. Montero, and I. M. Llorente. Evaluating the Reliability of Computational Grids from the End User's Point of View. *Journal of Systems Architecture,*, 52(12):727–736, 2006.

[HML06b] Eduardo Huedo, Ruben S. Montero, and Ignacio M. Llorente. A Modular Architecture for Interfacing Pre-WS and WS Grid Resource Management. *Future Generation Computing Systems*, 23(2):252–261, 2006.

[HSV+] Pilar Hernandez, Xavier Sole, Joan Valls, Victor Moreno, Gabriel Capella, Ander Urruticoechea, and Miguel Pujana. Integrative Analysis of a Cancer Somatic Mutome, journal = Molecular Cancer, volume = 6, pages = 13, year = 2007,.

[HWS+06] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn. Taverna: A Tool for Building and Running Workflows of Services. *Nucleic Acids Research*, 34(Web Server issue), 2006.

[IOP99] M. A. Iverson, F. Ozguner, and L. C. Potter. Statistical Prediction of Task Execution Times through Analytic Benchmarking for Scheduling in a Heterogeneous Environment. In *Proc. 8th Heterogeneous Computing Workshop (HCW'99)*, pages 99–111. IEEE Computer Society, 1999.

[KA00] Y-K. Kwok and I. Ahmad. Link Contention-Constrained Scheduling and Mapping of Tasks and Messages to A Network of Heterogeneous Processors. *J. Cluster Computing*, 3(2):113–124, 2000.

[KMH+06] Hartmut Kaiser, Andre Merzky, Stephan Hirmer, Gabrielle Allen, and Edward Seidel. The SAGA C++ Reference Implementation: A Milestone Toward New High-level Grid Applications. In *Proc. 2006 ACM/IEEE Conf. on Supercomputing (SC '06)*, page 184, 2006.

[KPD+04] Stefan Kurtz, Adam Phillippy, Arthur L. Delcher, Michael Smoot, Martin Shumway, Corina Antonescu, and Steven L. Salzberg. Versatile and Open Software for Comparing Large Genomes. *Genome Biology*, 5:R12, 2004.

[KS05] Peter Kacsuk and Gergely Sipos. Multi-Grid, Multi-User Workflows in the P-GRADE Grid Portal. *Journal of Grid Computing*, 3(3-4):221–238, 2005.

[LARS01] Gerd Lanfermann, Gabrielle Allen, Thomas Radke, and Edward Seidel. Nomadic Migration: A New Tool for Dynamic Grid Computing. In *Proc. of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC '01)*, page 429. IEEE Computer Society, 2001.

[LG06] Weizhong Li and Adam Godzik. CD-HIT: A Fast Program for Clustering and Comparing Large Sets of Protein or Nucleotide Sequences. *Bioinformatics*, 22:1658–1659, 2006.

[Liu07] Yihui Liu. Feature Extraction for Mass Spectrometry Data. In *Proc. 2007 Intl. Conf. on Life System Modeling and Simulation (LSMS'07)*, volume 4689 of *Lecture Notes in Computer Science*, pages 188–196, 2007.

[LMH05] Ignacio M. Llorente, Ruben S. Montero, and Eduardo Huedo. A Loosely Coupled Vision for Computational Grids. *IEEE Distributed Systems Online*, 6(5), 2005.

[LPSF77] A. G. Litvak, G. V. Permitin, E. V. Suvorov, and A. A. Fraiman. Electron-Cyclotron Heating of Plasma in Toroidal Systems. *Nuclear Fusion*, 17:659–665, 1977.

[MB05] T. Ma and R. Buyya. Critical-Path and Priority Based Algorithms for Scheduling Workflows with Parameter Sweep Tasks on Global Grids. In *Proc. 17th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2005)*, pages 251–258. IEEE CS, 2005.

[MFF+05] Anna Morajko, Enol Fernandez, Alvaro Fernandez, Elisa Heymann, and Miquel Angel Senar. Workflow Management in the CrossGrid Project. In *Proc. European Grid Conference (EGC2005)*, volume 3470 of *Lecture Notes in Computer Science*, pages 424–433, 2005.

[MHL06] R. S. Montero, E. Huedo, and I. M. Llorente. Benchmarking of High Throughput Computing Applications on Grids. *Parallel Computing*, 32(4):267–279, 2006.

[MHV+02] Marco A Méndez, Christian Hödar, Chris Vulpe, Mauricio González, and Verónica Cambiazo. Discriminant Analysis to Evaluate Clustering of Gene Expression Data. *FEBS Letters*, 522(1-3):24–8, 2002.

[NRW04] Jason Novotny, Michael Russell, and Oliver Wehrens. GridSphere: a Portal Framework for Building Collaborations: Research Articles. *Concurrency and Computation: Practice & Experience*, 16(5):503–513, 2004.

[OBS99] Michael A. Olson, Keith Bostic, and Margo I. Seltzer. Berkeley DB. In *USENIX Annual Technical Conference, FREENIX Track*, pages 183–191, 1999.

[OVY02] B. Olivier, B. Vincent, and R. Yves. The Iso-Level Scheduling Heuristic for Heterogeneous Processors. In *Proc. 10th Euromicro Conf. Parallel, Distributed and Network-based Processing (PDP 2002)*, pages 335–350. IEEE CS, 2002.

[Par08] Laxmi Parida. *Pattern Discovery in Bioinformatics. Theory and Applications*. Chapman & Hall/CRC Mathematical & Computational Biology, 2008.

[PCM+07] Lina Peng, K. Seluk Candan, Christopher Mayer, Karamvir S. Chatha, and Kyung Dong Ryu. Optimization of Media Processing Workflows with Adaptive Operator Behaviors. *Multimedia Tools and Applications*, 33(3):245–272, 2007.

[Pin02] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Prentice Hall, second edition, 2002.

[PLL+03] Steven T. Peltier, Abel W. Lin, David Lee, Stephen Mock, Stephan Lamont, Tomas Molina, Mona Wong, Lu Dai, Maryann E. Martone, and Mark H. Ellisman. The telescience portal for advanced tomography applications. *Journal of Parallel and Distributed Computing*, 63(5):539–550, 2003.

[RBC+03] H. Rajic, R. Brobst, W. Chan, F. Ferstl, J. Gardiner, J.P. Robarts, A. Haas, B.Ñitzberg, and J. Tollefsrud. Distributed Resource Management Application API Specification 1.0. Technical report, The Global Grid Forum, 2003.

[RBLD02] A. K. Ram, A. Bers, and C.Ñ. Lashmore-Davies. Emission of Electron Bernstein Waves in Plasmas. *Physics of Plasmas*, 9(2):409–418, 2002.

[RC04] D. Robinson and K. Coar. RFC3875: The Common Gateway Interface (CGI) Version 1.1. Technical report, IETF, 2004. Disponible en http://www.ietf.org/rfc/rfc3875.txt.

[RMSB06] Morris Riedel, Roger Menday, Achim Streit, and Piotr Bala. A DRMAA-Based Target System Interface Framework for UNICORE. In *Proc. 12th Intl. Conf. on Parallel and Distributed Systems (ICPADS '06)*, pages 133–138, 2006.

[SL93] G.C. Sih and E.A. Lee. A Compiler-Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures. *IEEE Transactions on Parallel and Distributed Systems*, 4(2):175–186, 1993.

[SNM+02] Keith Seymour, Hidemoto Nakada, Satoshi Matsuoka, Jack Dongarra, Craig Lee, and Henri Casanova. Overview of GridRPC: A Remote Procedure Call API for Grid Computing. In *Proc. 3rd Intl. Work. on Grid Computing (GRID '02)*, pages 274–278, 2002.

[SNW+07] Oliver G. Staadt, Vijay Natarajan, Gunther H. Weber, David F. Wiley, and Bernd Hamann. Interactive Processing and Visualization of Image Data for Biomedical and Life Science Applications. *BMC Cell Biology*, 8(1):10, 2007.

[SS94] S. Selvakumar and C. Siva Ram Murthy. Scheduling Precedence Constrained Task Graphs with Non-Negligible Intertask Communication onto Multiprocessors. *IEEE Transactions on Parallel and Distributed Systems*, 5(4):328–336, 1994.

[Sta81] W.M. Jr. Stacey. *Fusion Plasma Analysis*. Wiley-Interscience, 1981.

[THW02] H. Topcuoglu, S. Hariri, and M-Y. Wu. Performance-Effective and Low-complexity Task Scheduling for Heterogeneous Computing. *IEEE Transactions on Parallel and Distributed Systems*, 13(3):260–274, 2002.

[VD03] Sathish S. Vadhiyar and Jack J. Dongarra. A Performance Oriented Migration Framework For The Grid. In *Proc. of the 3st International Symposium on Cluster Computing and the Grid (CCGRID '03)*, page 130, 2003.

[vNMW+05] Rob V. van Nieuwpoort, Jason Maassen, Gosia Wrzesińska, Rutger F. H. Hofman, Ceriel J. H. Jacobs, Thilo Kielmann, and Henri E. Bal. Ibis: a Flexible and Efficient Java-based Grid Programming Environment: Research Articles. *Concurrency and Computation: Practice & Experience*, 17(7-8):1079–1107, 2005.

[VPHM+07] J. L. Vázquez-Poletti, E. Huedo, R. S. Montero, I. M. Llorente, A. Cappa, M. Tereshchenko, and F. Castejón. MARATRA: A Production Fusion Physics System. In *Proc. 2nd EGEE User Forum*, 2007.

[VPHML06] J. L. Vázquez-Poletti, Eduardo Huedo, Ruben S. Montero, and Ignacio M. Llorente. Coordinated Harnessing of the IRISGrid and EGEE Testbeds with GridWay. *Journal of Parallel and Distributed Computing*, 66(5 (IPDPS'04 Special Issue)):763–771, 2006.

[WF95] Owen White and Will FitzHugh. A Rapid Retrieval Tool for Operating on Large, Flat Archive Files. *Bioinformatics*, 11(1):45–47, 1995.

[YB05] Jia Yu and Rajkumar Buyya. A Taxonomy of Workflow Management Systems for Grid Computing. *J. Grid Computing*, 3(3–4):171–200, 2005.

[Yua05] Zheng Yuan. Better Prediction of Protein Contact Number Using a Support Vector Regression Analysis of Amino Acid Sequence. *BMC Bioinformatics*, 6:248–257, 2005.